applied sciences

MDPI

Article

Automatic Ontology-Based Model Evolution for Learning Changes in Dynamic Environments

Roua Jabla [1,2,*], Maha Khemaja [3], Félix Buendia [1] and Sami Faiz [4]

1 Department of Computer Engineering, Universitat Politècnica Valencia, Camino de Vera S/N, 46022 Valencia, Spain; fbuendia@disca.upv.es
2 ISITCom, University of Sousse, Sousse 4011, Tunisia
3 PRINCE Research Lab, ISITCom, University of Sousse, Sousse 4011, Tunisia; maha_khemaja@yahoo.fr
4 LTSIRS Laboratory, University of Tunis El Manar, Tunis 5020, Tunisia; sami.faiz@isamm.uma.tn
* Correspondence: jabla.roua@gmail.com

Abstract: Knowledge engineering relies on ontologies, since they provide formal descriptions of real-world knowledge. However, ontology development is still a nontrivial task. From the view of knowledge engineering, ontology learning is helpful in generating ontologies semi-automatically or automatically from scratch. It not only improves the efficiency of the ontology development process but also has been recognized as an interesting approach for extending preexisting ontologies with new knowledge discovered from heterogenous forms of input data. Driven by the great potential of ontology learning, we present an automatic ontology-based model evolution approach to account for highly dynamic environments at runtime. This approach can extend initial models expressed as ontologies to cope with rapid changes encountered in surrounding dynamic environments at runtime. The main contribution of our presented approach is that it analyzes heterogeneous semi-structured input data for learning an ontology, and it makes use of the learned ontology to extend an initial ontology-based model. Within this approach, we aim to automatically evolve an initial ontology-based model through the ontology learning approach. Therefore, this approach is illustrated using a proof-of-concept implementation that demonstrates the ontology-based model evolution at runtime. Finally, a threefold evaluation process of this approach is carried out to assess the quality of the evolved ontology-based models. First, we consider a feature-based evaluation for evaluating the structure and schema of the evolved models. Second, we adopt a criteria-based evaluation to assess the content of the evolved models. Finally, we perform an expert-based evaluation to assess an initial and evolved models' coverage from an expert's point of view. The experimental results reveal that the quality of the evolved models is relevant in considering the changes observed in the surrounding dynamic environments at runtime.

Keywords: ontology; OWL; ontology learning; semantic analysis

Citation: Jabla, R.; Khemaja, M.; Buendia, F.; Faiz, S. Automatic Ontology-Based Model Evolution for Learning Changes in Dynamic Environments. *Appl. Sci.* **2021**, *11*, 10770. https://doi.org/10.3390/app112210770

Academic Editor: Diana Kalibatiene

Received: 20 October 2021
Accepted: 12 November 2021
Published: 15 November 2021

1. Introduction

Ontologies are playing an increasingly important role in knowledge representation. In general, they provide a structured knowledge representation in which concepts are stored together with their properties and the relations between them [1]. However, ontologies can reflect inaccuracies in dynamic environments due to the fact that they are usually considered for environments with predictable behaviors. As a result, they are powerless to deal with unforeseen and new changes in the surroundings, since knowledge encapsulated in an ontology does not evolve over time. Thus, a considerable issue for using an ontology is the lack of support of a dynamic environment, in which changes are frequently emerging at runtime. To take into consideration dynamic environments, unforeseen changes have to be reflected in the ontologies at runtime. Therefore, ontologies must be used, and during their period of use, the knowledge on which they rely needs to be updated. Nevertheless, developing ontologies is deemed to be time-consuming and often expensive, but updating

them is even more expensive, which usually constitutes the potential limit for addressing the rapid changes occurring in dynamic environments.

One way to overcome the issue of irrelevant knowledge stored in ontologies within dynamic environments and the increasing costs associated with adapting them to environment changes is the use of an approach known as ontology learning. The ontology learning approach refers to the process of deriving relevant knowledge such as concepts, proprieties, and relations, as well as the occasional axioms from input data to build a new ontology or to enrich the preexisting one. That notwithstanding, ontology learning is not without issues due to the variety of changes that can arise in the enclosed environment at runtime. Hence, this cannot be manually performed effectively by experts, since they are not able to foresee all aspects of a change. Moreover, it also struggles to fully automate the learning process. Thus, most works in ontology learning require a minimum expert contribution and insight at runtime. For example, some works start building an ontology semi-automatically from scratch [2,3], requiring expert intervention at runtime. Furthermore, other works start enriching preexisting ontologies from scratch [4,5] based on expert contributions. To address these issues, we focus on proposing an automatic ontology-based model evolution approach to support dynamic environments at runtime through the evolution of an initial model expressed as an ontology. The aim of this proposed approach is twofold: (1) to analyze heterogeneous semi-structured input data for learning an ontology and (2) to use the learned ontology for extending and improving an initial ontology-based model to accommodate the changes that occur in its closed dynamic environment. The process of ontology-based model evolution is driven by the task of ontology learning and proceeds automatically with a non-expert contribution at runtime to deal with environment changes. In addition, we cover the implementation details and a case study to illustrate the proposed approach. Finally, we perform feature-based, criteria-based, and expert-based evaluation approaches to assess the quality of the proposed approach's results.

The remainder of the paper is structured as follows. Section 2 discusses related work and highlights our contributions. In Section 3, we present in detail an ontology-based model evolution approach. Section 4 describes a proof-of-concept prototype implementation and a case study. A validation of the proposed approach is given through evolved ontology-based model evaluation, which is introduced in Section 5. Section 6 presents the evaluation results to demonstrate the potential of the evolved ontology-based models. In Section 7, we conduct discussions about the main results achieved. Finally, Section 8 concludes with directions toward future work.

2. Related Work

In this section, we take a look at previous works that were interested in ontology learning approaches. Then, we end this section with a brief description of a comparative study.

2.1. Ontology Learning Approaches

For dynamic ontology-based model evolution, ontology learning approaches can offer evolution support. Ontology learning is defined as the automatic or semi-automatic process for building an ontology from scratch, enriching or adapting a preexisting ontology to the arisen changes, and the consistent management of these changes [6]. By going through the literature and several surveys in this area [7–9], numerous approaches have been proposed to support ontology learning in the scope of ontology building or preexisting ontology evolution.

A couple of approaches for ontology learning have been proposed in the scope of building ontologies from scratch. In this sense, Jablonski et al. [10] introduced an approach that deals with learning ontologies from tabular data (TD) in CSV format. The proposed approach relies on two-level data transformation from the CSV data to a "pivot" model and then to an ontology. To this end, a mapping process is introduced to first transform a CSV document into an XML format. Then, the transformation from the XML format to the ontology is performed using Extensible Stylesheet Language Transformation (XSLT). We

also cite the work of Völker and Niepert [11], which described an approach for ontology building directly from DBpedia. The presented approach uses statistical methods and association rules for finding alignments. The novelty of this approach comes from the fact that it is one of the early works which creates ontologies from a Linked Open Data (LOD) dataset. Another work that has considered ontology building based on ontology learning is that of Bohring and Auer [12]. This work proposed a tool called xml2owl, which covers both the schema and data levels. Xml2owl offers the possibility to create an OWL ontological model from an XML schema and to convert XML data to OWL instances. This conversion is performed using XSLT. Moreover, Krataithong et al. [3] proposed a semi-automatic framework that provides a support system for users in transforming TD in spreadsheet format (i.e., CSV or XSL to RDF format). This framework focused on the need for automatic schema detection before starting ontology learning from TD. For that, it provided techniques for data type and nominal type detection for the column fields in the datasets. Therefore, the proposed approach consists of three processes in which users are involved: dataset preparation and import, schema detection and verification, and OWL and RDF data generation. The last process relies on a pivot model that translates TD into a Relational Database (RDB) and mapping rules for database schema to ontology transformation. Aside from this, the authors in [13] proposed an approach to create an ontology from an RDB automatically. First, the presented approach performs relational schema pre-processing on the input data to determine its components. Then, it analyzes these components using a set of mapping rules to convert them to corresponding ontology components. Finally, Sbai et al. [14] proposed a semi-automatic approach to learn OWL ontologies from RBD by using classification techniques or, more specifically, decision trees. They introduced an algorithm based on the use of decision trees to automatically discover the taxonomic relationships between concepts. Then, they invited a domain expert to semantically validate the generated schema and add a meaningful name to the ontology concepts.

Apart from these approaches, other approaches in the scope of preexisting ontology evolution are emerging. For example, Yao et al. [4] proposed a mechanism to transform semi-structured data—specifically a set of JSON documents provided by Web services—into a unified ontology. This mechanism extracts JSON data automatically, including concepts, properties, constraints, and values and builds the ontology. The resulting ontology must be validated by domain experts. Additionally, Booshehri and Luksch [5] provided an ontology enrichment approach from text, in which the Web of linked data, particularly DBpedia, is used. They explored DBpedia as background knowledge alongside the text in order to discover implicit knowledge regarding the text, from which new ontological relations, specifically object properties, were inferred. The proposed approach aimed at recommending only new object properties to ontology engineers, enabling them to create much more expressive ontologies. This follows previous works in the field of ontology learning from text with the difference that it also uses the knowledge scattered in DBpedia to improve the ontology learning output. Moreover, Aggoune [15] introduced a semantic approach for automatic ontology learning from many heterogeneous RDBs in order to facilitate their integration. This approach uses WordNet as a lexical database and a semantic similarity measure to help select the best terms to represent the ontology components. Finally, Sbissi et al. [16] proposed an approach based on automatic ontology learning from unstructured text to evolve a preexisting ontology. The process of ontology learning starts with the analysis of the text. Then, it switches to the extraction of relevant terminology synonymous with the identification of terms, concepts, concept hierarchy organization, relationships, and extraction of axioms. Once these elements are extracted, it updates the preexisting ontology by adding them.

2.2. Comparison Criteria

To provide a comparative study of the discussed approaches, we introduce different criteria, with some of them arising from previous studies [17,18], as follows:

- Input. Different inputs are used to learn ontologies. They are grouped as structured, semi-structured, and unstructured input data;
- Learning element. Learning elements are classified as follows: concept (C), taxonomic relation (TR), non-taxonomic relation (NTR), datatype property (D), individual (I), and axiom (A);
- Pivot model. This is a model that behaves as a unification of heterogeneous inputs by transforming them to an intermediate model during the process of ontology learning;
- Pivot model's hierarchy. A pivot model can detect and maintain the underlying hierarchical structure in the input data;
- Ontology Refinement. This is achieved through improving learned ontologies, for example, by discovering and integrating new relations using different external resources;
- Ontology Alignment. This is achieved using a process of determining correspondences between terms in a preexisting ontology and a learned ontology;
- Ontology Merging. This is achieved by extending a preexisting ontology through the addition of new concepts, relations, properties, individuals, and axioms;
- Automation Degree. The acquisition of knowledge may be performed automatically or semi-automatically, where it is handled with the help of users or experts.

2.3. Comparison and Discussion

Based on the aforementioned criteria, we classified the ontology learning approaches as shown in Table 1.

Table 1. Tabular comparison of different ontology learning approaches based on main distinguishing criteria.

	Input	Learning Element	Pivot Model	Pivot Model's Hierarchy	Ontology Refinement	Ontology Alignment	Ontology Merging	Automation Degree
Jablonski et al. [10]	Semi-Structured: CSV	I	XML data	x	x	x	x	Automatic
Völker and Niepert [11]	Strucured: DBpedia	C + NTR + I	x	x	x	x	x	Automatic
Bohring and Auer [12]	Semi-Structured: XML	C + NTR + D + I + A	x	x	x	x	x	Automatic
Krataithong et al. [3]	Semi-Structured: CSV or XSL	C + NTR + D	RDB schema	x	x	x	x	Semi-Automatic
Lakzaei and Shmasfard [13]	Structured: RBD	C + TR + NTR + D + I + A	x	x	x	x	x	Automatic
Sbai et al. [14]	Structured: RBD	C + TR	x	x	x	x	x	Semi-Automatic
Yao et al. [4]	Semi-Structured: JSON	C + NTR + D + I	x	x	Predefined Rules	√	√	Semi-Automatic
Booshehri and Luksch [5]	Unstructured: Text	NTR	x	x	LOD with DBpedia	x	√	Semi-Automatic
Aggoune [15]	Structured: RBD	C + NTR + D + I	x	x	x	x	√	Automatic
Sbissi et al. [16]	Unstructured: Text	C + TR + NTR + D+I + A	x	x	x	x	√	Automatic
Our approach	Semi-Structured: CSV, JSON or XML	C + TR + NTR + D + I + A	XML schema+ data	√	LOD with DBpedia + WordNet + Metadata	√	√	Automatic

√: Supported; x: Not supported.

The comparative table shows that a large number of approaches, such as those in [4,5,10–16], have a single input format. For example, Volker and Niepert [11] built an ontology from LOD (DBpedia). However, our approach is capable of handling a semi-structured input with varying formats, including CSV, JSON, and XML. In addition, almost all approaches aimed to return one or a few specific learning elements. For example, Booshehri and Luksch [5] extracted only non-taxonomic relations, while Aggoune [15] included four learning elements such as concepts, non-taxonomic relations, datatype properties, and individuals. In contrast to these common approaches, we cover the six different kinds of learning elements that can be appended to a learned ontology. Moreover, apart from the works of Jablonski et al. [10] and Krataithong et al. [3], where an input is transformed to an ontology on two levels using an intermediate XML data and RDB schema,

respectively, a one-level transformation was applied in the remaining approaches. In this work, similar to [10], we propose a two-level transformation, since we support heterogeneous formats of input. Therefore, a semi-structured input is transformed to an ontology on two levels, going through a pivot used as an intermediate model in order to represent all input formats in the same formalism. Transformation on the first level refers to the transformation of a semi-structured input to a pivot model in XML schema. On the second level, it refers to the transformation of the pivot model itself to an ontology. Moreover, the XML hierarchy was overlooked in the reviewed approaches that used XML as an input [12] or as a pivot model [10]. These approaches tended to flatten out the XML hierarchical structure. As a result of the XML hierarchy's absence, the quality of the ontology learning results could worsen, since an ontology should have a hierarchy structure. To address this absence, we considered the XML hierarchy structure in pivot models by exploiting the hierarchy imposed by the XML language. Furthermore, none of the reviewed approaches dealt with ontology refinement through coupling several methods to answer the missing knowledge during ontology learning. They usually used DBpedia as in [5] or predefined rules as in [4] to extract missing knowledge for refining learned ontologies. Contrary to this, we spent more effort on improving the refinement phase. Therefore, LOD such as DBpedia, WordNet, and lexico-syntactic patterns in metadata will be applied as references to accomplish the refinement phase. In contrast to [11], we do not learn an ontology from LOD; however, we make use of LOD, WordNet, and metadata as background knowledge for refining a learned ontology. Finally, this comparative study highlights that the previously discussed approaches in the scope of preexisting ontology evolution [4,5,12,13] did not consider the ontology alignment, in contrast to the ontology merging. Indeed, close to [4], we consider an evolution process that is driven by the task of ontology learning. The proposed evolution process includes alignment and merging activities to consider the evolution of an initial ontology-based model using learning elements resulting from the learning process.

Following the above discussion, we aim to close gaps within the related works by proposing an ontology-based model evolution approach that follows ontology learning to evolve an initial ontology-based model to answer the changes encountered in dynamic environments at runtime. In addition, we take into consideration the requirement for hierarchical XML pivot models to reorganize and improve the learned ontology representation. Furthermore, we focus on improving the ontology refinement phase, which relies on using different external background knowledge such as DBpedia, WordNet, and metadata. To sum up, the main strengths of the proposed approach are fivefold: (1) the fully automatic ontology-based model evolution guided by ontology learning once changes in the surrounding environments occur at runtime; (2) the support of heterogeneous input data; (3) the use of a hierarchical XML pivot model to maintain the input's hierarchy and to construct the learned ontology's hierarchy; (4) the use of multiple pieces of background knowledge together with a pattern-based semantic analysis, including simple patterns such as regular expressions to ensure the refinement of the learned ontologies; and (5) the neglect of expert intervention.

3. Ontology-Based Model Evolution Approach

This section presents an ontology-based model evolution approach based on ontology learning. The main challenge of this approach is the automatic evolution of a model at runtime to deal with the continuous changes of or within the surrounding dynamic environment. To meet this challenge, it is worth considering the idea of ontology modularity, where semantic enrichment and evolution come into play. Here and within the modularity notion, the model grounded on ontology could semantically be enriched in a flexible way across the runtime environment changes. The major objective of the proposed approach is to extend an initial modular, ontology-based model through automatically learning ontology from a semi-structured data source and then enriching and populating it to conclude in answering to the runtime environment changes. We pursue evolving an

initial ontology-based model through transforming a semi-structured data source to an ontology, since the data sources could well represent the changes arising in the enclosing environments rather than the external knowledge sources, such as, DBpedia or WordNet. To exhibit a strong evolution, our proposed ontology-based model evolution approach will not be limited to these external knowledge sources and will rely on the use of semi-structured data sources, which can be far more extensive and representative for answering dynamic environments. For fulfilling an ontology-based model evolution, the presented approach, as illustrated in Figure 1, involves several modules, such as data source selection, data source format unification, ontology-based model learning, and ontology-based model integration modules.

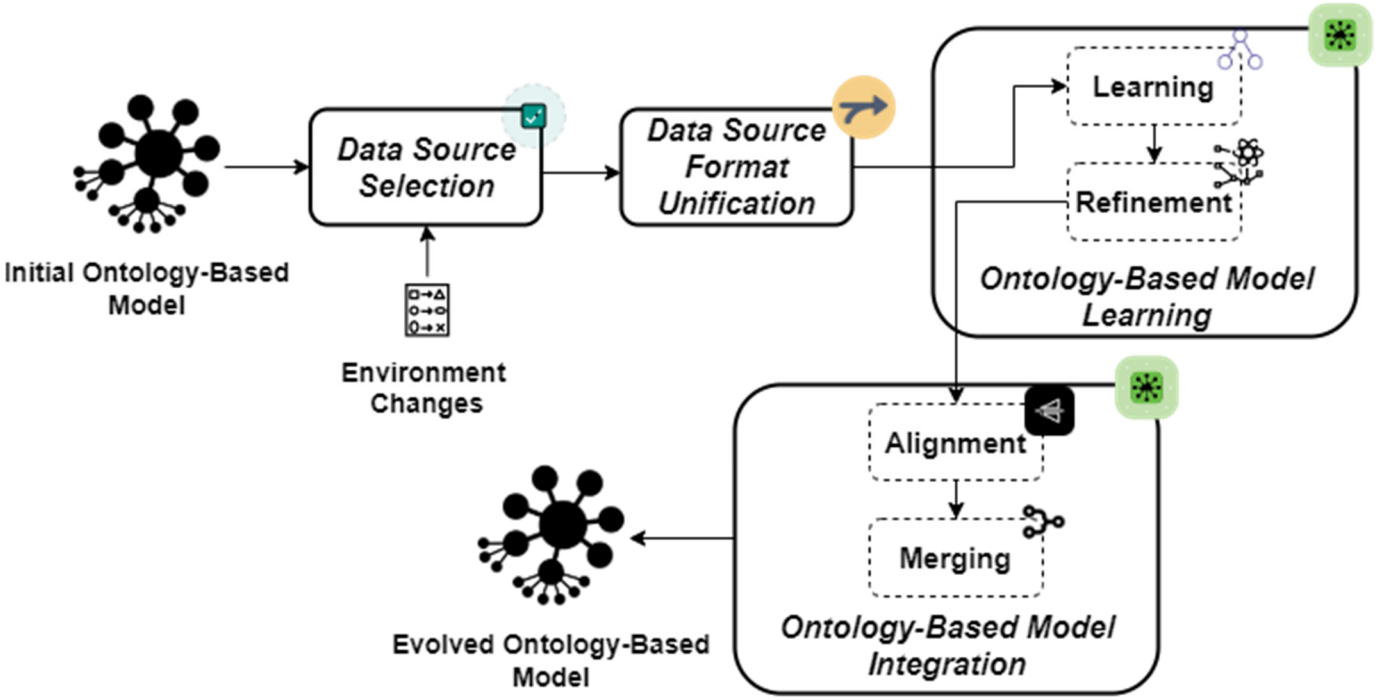


Figure 1. Ontology-based model evolution approach.

3.1. Data Source Selection Module

Once changes occur in the surrounding environment, the data source selection module aims to retrieve and select a candidate semi-structured data source that can cope with the captured changes. The candidate data source should be accompanied by its metadata to go further in the refinement of the learned ontology. The use of metadata is needed to increase the quality of the learned ontology by finding frequently occurring patterns and recognizable structures within the textual description of the candidate data source, since a semi-structured data source may suffer from a slight lack of semantics.

The present module is handled by an automated data source search engine. Figure 2 illustrates a simple and general view of three main components of this search engine. The indexer component is in charge of parsing data sources and collected surrounding environment changes to be more efficient and accurate for performing retrieval techniques. It usually performs lowercase transformation, the removal of stop words, and stemming. Then, it stores parsed data sources in indexes. The query engine component is responsible for formulating a query from the parsed environment changes and then running it on the previously built indexes to get a list of the appropriate data sources that can match with the changes according to term frequency. The relevance ranker component is in charge of ranking the retrieved list to select the most useful data source regarding the environment changes. It first arranges the appropriate data sources by comparing their term frequencies and then selects the data source with the highest frequency.

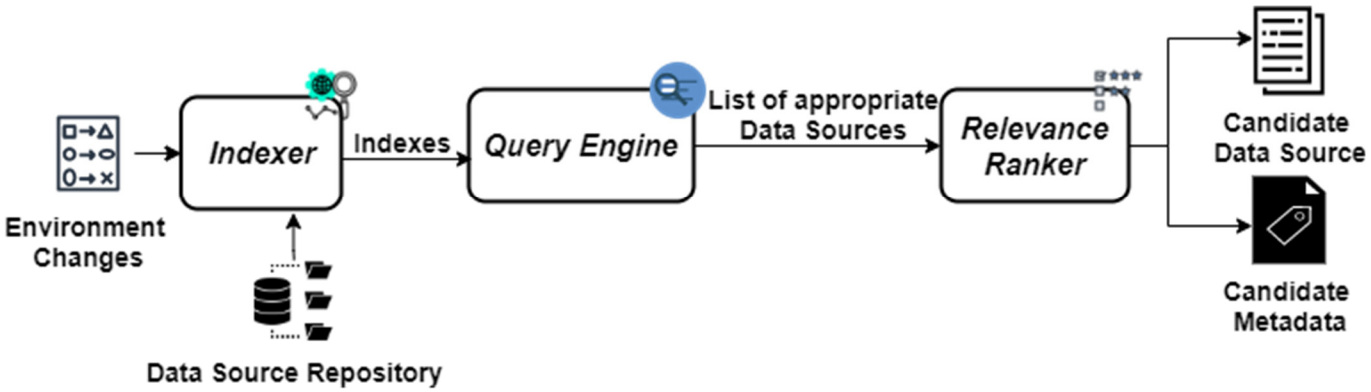


Figure 2. Data source search engine.

3.2. Data Source Format Unification Module

As we consider heterogeneous semi-structured formats, the main purpose of the data source unification module is to transform any semi-structured candidate data source to hierarchical XML data as a pivot model in accordance with the first-level transformation. Therefore, after selecting the candidate data source, the preprocessing step is started. This step takes the data source in question as the input and stops when it is cleaned and arranged in a structured form. Then, the underlying module behaves differently in each input format. For the TD, the arranged data are being parsed in order to automatically identify their schema. Then, a deep visit is performed on the parsed data to map them to the corresponding XML data by choosing an appropriate mapping rule for each tabular element and generating the corresponding XSD element. In order to achieve this, we defined a set of mapping rules, detailed in Table 2, for transforming TD, specifically CSV data, to hierarchical XML data, since existing libraries and tools map a TD to flat XML data, which can give rise to difficulties while transforming XML data to an ontology. For the JSON data, the arranged data are being transformed to the corresponding XML data upon an API-based mapping. For the XML data, the arranged data are analyzed to infer the related XML schema representation. At the end, this module produces as the output an XML document along with an XML schema document to be served as the input for the ontology-based model's learning module.

Table 2. CSV to XML schema mapping rules.

CSV	XML Schema
Table name (group of CSV files)	Complex type
Column (non-nominal datatype)	Attribute with column datatype as type
Column (nominal datatype)	Element inside an anonymous complex type
Column (group of CSV files)	Element inside an anonymous complex type
Label column	Complex type uses extension

3.3. Ontology-Based Model's Learning Module

After performing the first level-transformation, the ontology-based model learning module takes place to apply the second-level transformation and different refinement methods. Thus, this module starts with the definition of a local ontology in the OWL language, which is discovered from the XML pivot model, and finishes with some refinements in the local ontology. Hence, both phases are involved: the learning phase and the refinement phase.

3.3.1. Learning Phase

The learning phase deals with the acquisition of information needed to learn a local ontology starting from the XML pivot model. The beginning step in this phase is intended

to parse the earlier generated XML schema for inspecting its features, such as complex types, elements, attributes, and so on. Then, the parsed schema results are traversed to transform the pivot model to an OWL local ontology. The implementation of the following transformation adheres to certain mapping rules, which are introduced in Table 3. These rules determine how to convert each feature of the XML schema to the corresponding ontology element. Once this transformation is established, the data carried by the XML document is transformed to the ontology individuals.

Table 3. XML schema to OWL ontology mapping rules.

XML Schema	OWL Ontology
xsd:complexType: xsd:element, containing other elements or having at least one attribute	owl:Class, coupled with owl:ObjectProperty
xsd:attribute	owl:DataProperty with a range depending on the attribute type
xsd:element, inside an anonymous complex type	owl:ObjectProperty
xsd:complexType, which uses extensions	owl:Class as an owl:subClassOf "base type"

3.3.2. Refinement Phase

After attaining the local ontology, it is necessary to expand its quality borders through the accreditation of a refinement phase. Therefore, uncovering missing concepts as well as relations and identifying a concept hierarchy are considered ontology refinement methods. For this, DBpedia, WordNet, and metadata knowledge bases are considered references for refining the local ontology, since they are widely used and cover broad resources from different domains. Indeed, coupling several knowledge bases could answer missing knowledge during ontology refinement and improve the inexpressive local ontology through exploiting the different linguistic patterns, hyponym-hypernym relationships contained in WordNet, and concept hierarchy retrieved from DBpedia.

The first refinement method, "missing concepts refinement", is used to compensate for the missing concepts from the learning phase. More specifically, this method follows the main steps outlined in Algorithm 1. In this algorithm, the beginning step is intended to collect all datatype properties included in the local ontology. Next, the retrieved datatype properties are traversed to extract their corresponding domains. Then, the candidate metadata are processed for each datatype property sequentially in three steps. First, each sentence that contains the datatype property is extracted. Second, Parts of Speech (POS) tags are induced over each extracted sentence to prepare them for analysis. Third, a collection of lexico-syntactic patterns (LSPs), presented in Table 4, is exploited over the POS tags to extract the related domain to the datatype property. These LSPs are issued from different works published in the literature, such as [19–21], to perform a semantic analysis on any textual metadata and to identify missing concepts through the datatype properties. Once these steps are established, the domain of the datatype property is updated to represent the missing concepts.

The second refinement method, "taxonomic relation refinement", identifies the concept hierarchy to organize learned and new covered concepts into the local ontology through subsumption relations. Therefore, this method focuses on exploring the benefits from coupling WordNet, metadata, and DBpedia to learn new taxonomic relations among the concepts represented in the partially refined local ontology.

Algorithm 1. The Proposed Missing Concepts Refinement Algorithm

Input: LocalOntology, Metadata, LSPs
Output: PartiallyRefined_LocalOntology
1. DataTypePropertiesSet = GetAllDatatypeProperties (LocalOntology)
2. **For each** DatatypeProperty ∈ DataTypePropertiesSet **do**
3. Domain = GetDomain (DatatypeProperty)
4. SentencesSet = FindSentences (DatatypeProperty, Metadata)
5. **For each** Sentence ∈ SentencesSet **do**
6. **Sentence = PartsOfSpeechTagging(Sentence, POS)**
7. NewDomain = matches (Sentence, LSPs)
8. **If** NewDomain ≠ ∅ **then**
9. PartiallyRefined_LocalOntology = UpdateMissingConcept (DatatypeProperty,Domain,NewDomain)
10. **End If**
11. **End For**
12. **End For**
13. **End.**

Table 4. LSPs for missing concept acquisition.

LSP	Example	Relation
Property | ies | characteristic | s | attribute | s of NP<class> be [PARA] [(NP<property>,) * and] NP<property>	Attributes of an accelerometer are X, Y, and Z	Datatype properties: X, Y, Z Class or domain: accelerometer
NP<class> be [(AP<property>,) *] and AP<property>	Metals are lustrous, malleable, and good conductors of heat and electricity	Datatype property: lustrous Class or domain: metal
NP<class> have NP<class>	A car has a color	Datatype property: color Class or domain: car
NN with | without DT? RB? JJ? ATTR	A pizza with some cheese.	Datatype property: cheese Class or domain: pizza
DT ATTR of DT? RB? JJ? NN	The color of the car	Datatype property: color Class or domain: car

NP: noun phrase; AP: adjectival phrase; NN: noun; DT: determiner; RB: adverb; JJ: adjective; ATTR: attribute; PARA: paralinguistic symbols, like colons; *: repetition.

Indeed, the most novel idea in this method is to compose several steps that maximize the performance of the concept hierarchy refinement by taking into consideration (1) the verb hyponym and hypernym relationships contained in WordNet, (2) the behavior of different linguistic patterns by extracting hyponym-hypernym pairs from the metadata accompanied by the candidate data source, and (3) the concept hierarchy retrieved from DBpedia predicates, such as rdfs:subClassOf, umbel:superClassOf, and geo-ont:parentFeature. Another possibility to improve the taxonomic relation refinement from the metadata is to combine the Hearst [22,23] and Aguado de Cea [19] patterns for hyponym and hypernym extraction, given that this combination has been proven to improve the precision and recall, since Hearst patterns allow for finding all possible taxonomic relationships with high precision but low recall, and in contrast, Aguado de Cea patterns produce high recall but low precision [24]. Table 5 shows a part of Hearst and Cea's patterns that are used to acquire new taxonomic relations in the partially refined local ontology.

The "taxonomic relation refinement" method is outlined in Figure 3, where the basic steps are displayed. At the initial step of this method, the candidate metadata is scanned for instances of distinguished Hearst and Cea patterns that are useful for detecting hyponym and hypernym relations for each concept included in the partially refined local ontology to identify new assumption relations. These patterns occurred frequently across the textual metadata and summarized the most common ways of expressing hyponyms and hypernyms. An example of these patterns that could be detected in a sentence like "Activities such as changing clothes, having guests, or cleaning are considered" is "NP {,} such as {NP,} * {and | or} NP", where the first NP denotes a super concept (e.g., "activity") of the next NPs (e.g., "changing clothes", "having guests", and "cleaning"). Then, a set of hyponyms

and hypernyms covered by WordNet is gathered for each concept. Finally, this method finishes by checking each concept for the DBpedia predicates, such as "rdfs:subClassOf", "umbel:superClassOf", or "geo-ont:parentFeature", to discover unrecognized assumption relations among concepts using DBpedia.

Table 5. Hearst and Cea's patterns for hyponym and hypernym relation extraction.

Pattern Group	Pattern
Hearst's patterns	NP {,} such as {NP,} * {and | or} NP NP {,} including {NP,} * {and | or} NP NP {,} especially {NP,} * {and | or} NP
Cea's patterns	[(NP<subclass>,) * and] NP<subclass> be [CN] NP<superclass> [(NP<subclass>,) * and] NP<subclass> (classify as) | (group in | into | as) | (fall into) | (belong to) [CN] NP<superclass> There are CD | QUAN [CN] NP<superclass> PARA [(NP<subclass>,) * and] NP<subclass>

NP: noun phrase; CN: class name; CATV: verbs of classification; PARA: paralinguistic symbols like colons; *: repetition.

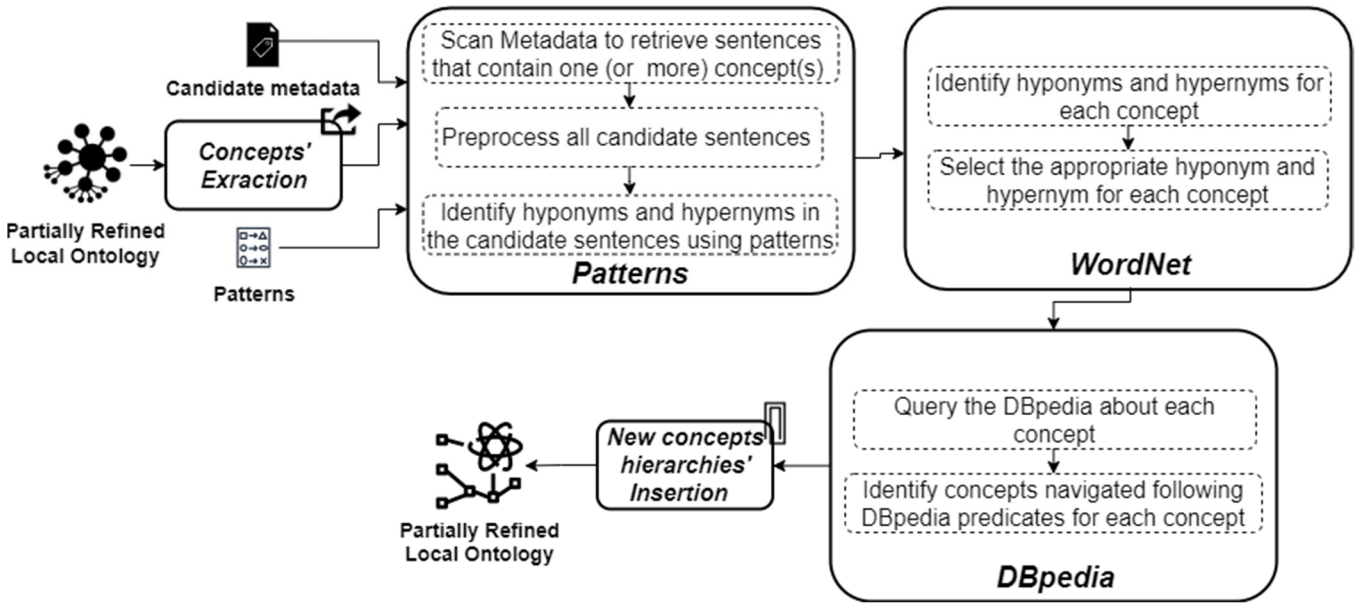


Figure 3. Taxonomic relation refinement method.

The third and last refinement method, "non-taxonomic relation refinement", is based on associating concepts by representing hidden connections between them and identifying their non-taxonomic relations, specifically the object properties. This method focuses on investigating the DBpedia and the metadata as background knowledge for discovering and automatically labeling the non-taxonomic relationships from the candidate sentences to refine the inexpressive concepts in the partially refined local ontology, as depicted in Figure 4. Within this method, the DBpedia and the metadata are considered to find new non-taxonomic relations related to different concepts existing in the partially refined local ontology. First, the comments about every partially refined local ontology's concept founded under "rdfs:comment" are explored in DBpedia. Next, the metadata are looped to extract all the sentences containing any partially refined local ontology concept. Then, a semantic analysis of the candidate comments and sentences is performed to locate their main components, such as, nouns, verbs, and so on. For this purpose, the candidate comments and sentences are analyzed with the aid of POS tags to identify verbs that will be used to label non-taxonomic relations.

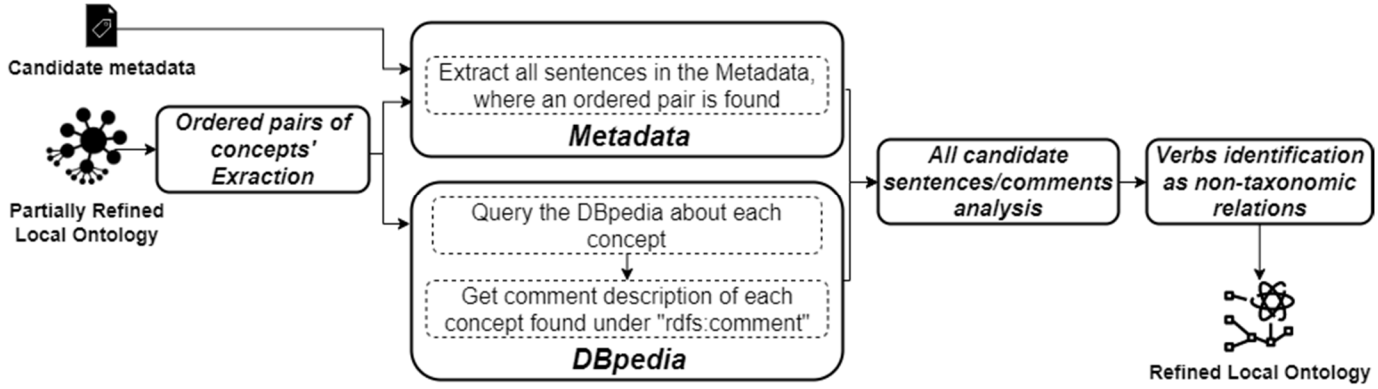


Figure 4. Non-taxonomic relation refinement method.

3.4. Ontology-Based Model Integration Module

The ontology-based model integration module is targeted toward updating an initial ontology-based model through a refined local ontology, which is constructed by analyzing a semi-structured data source that answers the arisen changes in the surrounding environment. This module is run every time the knowledge stored in the initial ontology-based model is not enough to fit with the arisen changes at runtime. To fulfill the integration, we take the two ontologies as inputs, and we carry out two important steps. First, we discover the correspondences between the different ontologies based on the similarity calculation. Second, we merge the new terms into the initial ontology-based model in order to obtain an evolved ontology-based model. To this end, we provide a twofold process involving an alignment phase and a merging phase.

3.4.1. Alignment Phase

To handle the ambiguity associated with formalization before the evolution of initial ontology-based model evolution, an alignment is essential to discover the correspondences between the refined local ontology and the initial ontology-based model. Therefore, the relations among the terms, which can be concepts, relations, properties, or individuals, should be mined before the merging activity as much as possible. For this purpose, we propose an automatic alignment approach consisting of two stages: an initial syntactic similarity measure followed by a semantic similarity measure. In the first stage, a simple distance computation between two strings labeling two terms is performed. Secondly, a semantic similarity is used to compute the extent of similarity between the term pairs regarding the likeliness of their meaning. This opens the door to exploring WordNet as a way of finding semantic similarities, since WordNet can determine the semantic distance between two terms' names by considering synonyms or the relation between the hypernym and hyponym. Given two terms T_1 and T_2, their similarity sim (T_1, T_2) is calculated according to the following equation:

$$sim(T_1, T_2) = \begin{cases} 1, if\ the\ term\ T_1\ \ is\ part\ of\ the\ synset\ of\ the\ term\ T_2\ \ or\ vice\ versa \\ 0.5\ if\ T_1\ \ is\ a\ hypernym\ or\ a\ hyponym\ of\ T_2\ \ or\ vice\ versa \\ 0, otherwise \end{cases}, \quad (1)$$

In this work, the combination of both syntactic similarity and the semantic similarity between terms intends to improve the overall alignment performance. At the end, the alignment activity provides a set of matches between the terms extracted from the refined local ontology and the initial ontology-based model according to the chosen strategy. This matching is defined in the form of triplets (T_1, T_2, r), where T_1 and T_2 are the terms of the two ontologies and r is a type of relation such as equality, generalization, or specialization.

3.4.2. Merging Phase

After the alignment phase, a merging phase is required to merge the refined local ontology with the initial ontology-based model. In our work, the merging phase is seen as the process that updates an initial ontology-based model through the addition of new terms such as concepts, properties, relations, and individuals from a refined local ontology to obtain a more complete ontology that can cover the environment's changes emerging at runtime. In other words, the initial ontology-based model is enriched and populated using the refined local ontology, taking into account the set of matchings defined in the previous phase. Based on the found alignments, similar terms are merged into a single one in the initial ontology-based model, whereas the terms considered dissimilar are directly copied into the initial ontology-based model. Finally, an evolved view of the initial ontology-based model is computed from the refined local ontology.

4. Proof of Concept and Case Study

In this section, we describe a prototypical implementation of the proposed approach. Furthermore, we present a case study to validate the proof of concept.

4.1. Proof of Concept

A proof of concept was implemented to verify the feasibility of the presented approach. In our proof of concept implementation, an application for assisting engineers was used to evolve users' initial ontology-based models regarding arisen changes in the users' surrounding environments at runtime. The presented application consisted of two distinct layers, called frontend and backend. The frontend layer consisted of an Angular-based web application for engineers, while the backend layer dealt with the automatic ontology-based model evolution. This layer was accessible from the frontend layer via RESTful Web services. All of the approach's modules were implemented as RESTful Web services, developed using Spring Boot and Java technologies. We adopted Spring Boot as the basic framework to simplify the work, since it is an open-source Java-based framework that makes the development of RESTful Web services simple.

To accomplish the backend layer implementation, we made use of Apache Lucene [25] to select the most useful semi-structured data source that could answer the changes occurring in the user's surrounding environment. We opted for Apache Lucene, as it is a powerful information retrieval tool that provides Java-based indexing and search technology. For the data source format unification, the set of mapping rules described in Table 2 was written in Java to transform the tabular data to XML data. In addition, a common JSON parsing API named Jackson, which converts JSON to XML data, and Trang API, which produces XML schema from an XML data input, were used. For the learning phase, an XML-Schema Object Model (XSOM) parser was utilized to parse the generated XML schema and inspect the elements and attributes in it. Moreover, the set of mapping rules defined in Table 3, together with the Jena API, were integrated to build the local ontology and then fill it via concepts, relations, properties, and individuals. Then, for the refinement phase as well as the evolution module, we used the extended Java WordNet Library (extJWNL), DBpedia, and Matcher Java regex classes.

4.2. Case Study

A case study was conducted to show how the proof of concept behaved in a realistic setting. This case study considered two scenarios. The first scenario covered moving from an ordinary apartment to a smart apartment, and the second covered moving from a smart apartment to a smart home.

- Scenario 1: Moving from an ordinary apartment to a smart apartment.

In this scenario, the user lives in an ordinary apartment during the years of study (t_0). The surrounding environment in which he or she lives contained only sensors commonly found in his or her smartphones, as shown in Figure 5a. Obviously, the user's initial ontology-

based model t_0 answered to his or her surrounding environment as depicted in Figure 6. Then, after he or she graduated from the university, the user decided to move to a smart apartment equipped with sensors. At t_1, this new environment, described in Figure 5b, was visited by the user. As illustrated in Figure 5b, a range of sensors from magnetic to electric, flush, PIR, and pressure sensors were available in the new user's smart apartment.

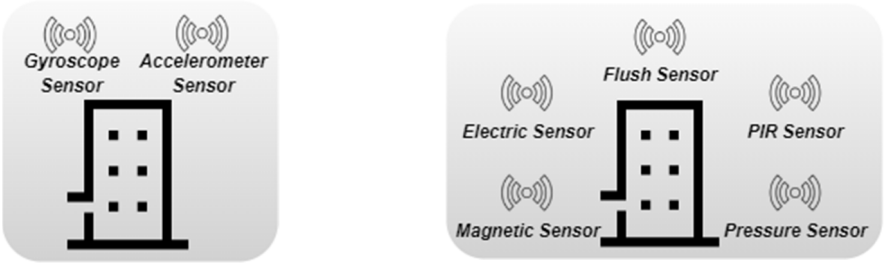


Figure 5. User's surrounding environments (**a**) at t_0 and (**b**) at t_1.

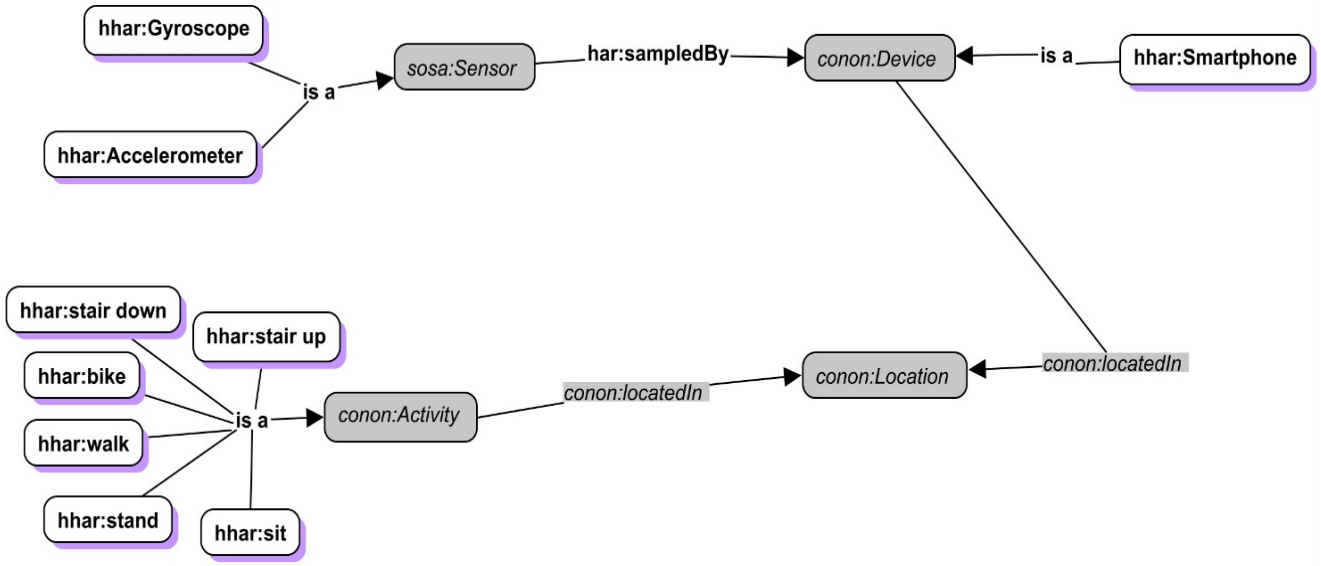


Figure 6. An excerpt of the user's initial ontology-based model at t_0.

Consequently, the initial ontology-based model, previously presented in Figure 6, would answer to the new environment illustrated in Figure 5b. To provide a strong answer, the data source selection service applied the automated data source search engine to select the appropriate data source using Apache Lucene. Figure 7 shows that it adopted the Ordonez dataset [26] for the evolution of the user's model at t_1. It contained 20,358 observations in CSV format conducted with 14 sensors. These sensors captured about 10 basic activities. The candidate dataset was accompanied by metadata, a sample fragment of which is depicted in Figure 8.

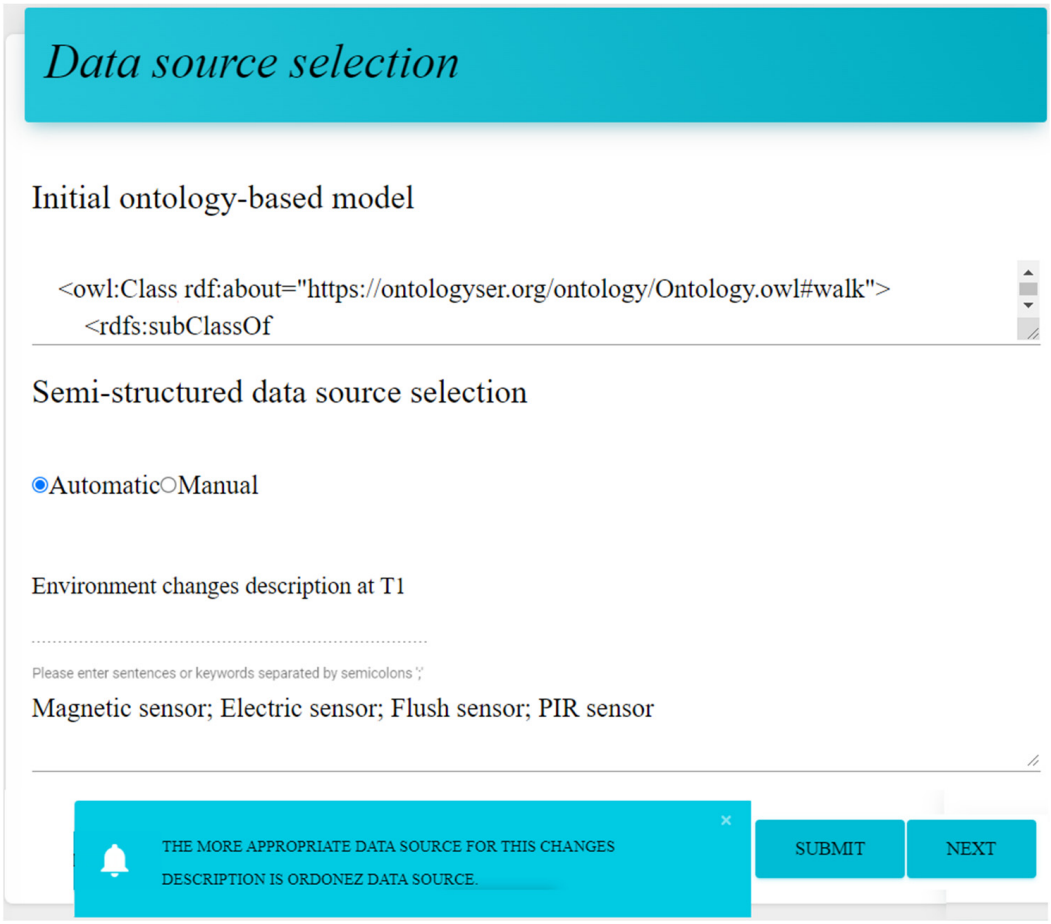


Figure 7. Candidate data source selection at t_1, the Ordonez dataset.

```
Ordonez  DATASET DESCRIPTION
============================
This dataset comprises information regarding the ADLs performed by two users on a daily basis
in their own apartments.
Smart apartment setting: 5 rooms
Activities (ADLs included): Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Dinner,
Snack, Spare_Time/TV, Grooming
Sensors:        PIR: Shower, Basin, Door Kitchen, Door Bathroom, Door Bedroom
                Magnetic: Maindoor, Fridge, Cupboard
                Flush: Toilet
                Pressure: Seat, Bed
                Electric: Microwave

SENSOR EXPLANATION
Sensor such as Magnetic, Flush, Pressure, Electric and PIR has place in the apartment.

ACTIVITY EXPLANATION
Leaving, toileting, showering, sleeping, breakfast, lunch, dinner, snack, spare_Time/TV
and grooming belong to activity.

These activities held in different places.
```

Figure 8. A sample fragment of the Ordonez metadata.

Next, the second and third modules used the Ordonez dataset together with the set of background knowledge to learn and refine a local ontology model. The output of the third module was the refined local ontology, an excerpt of which is shown in Figure 9.

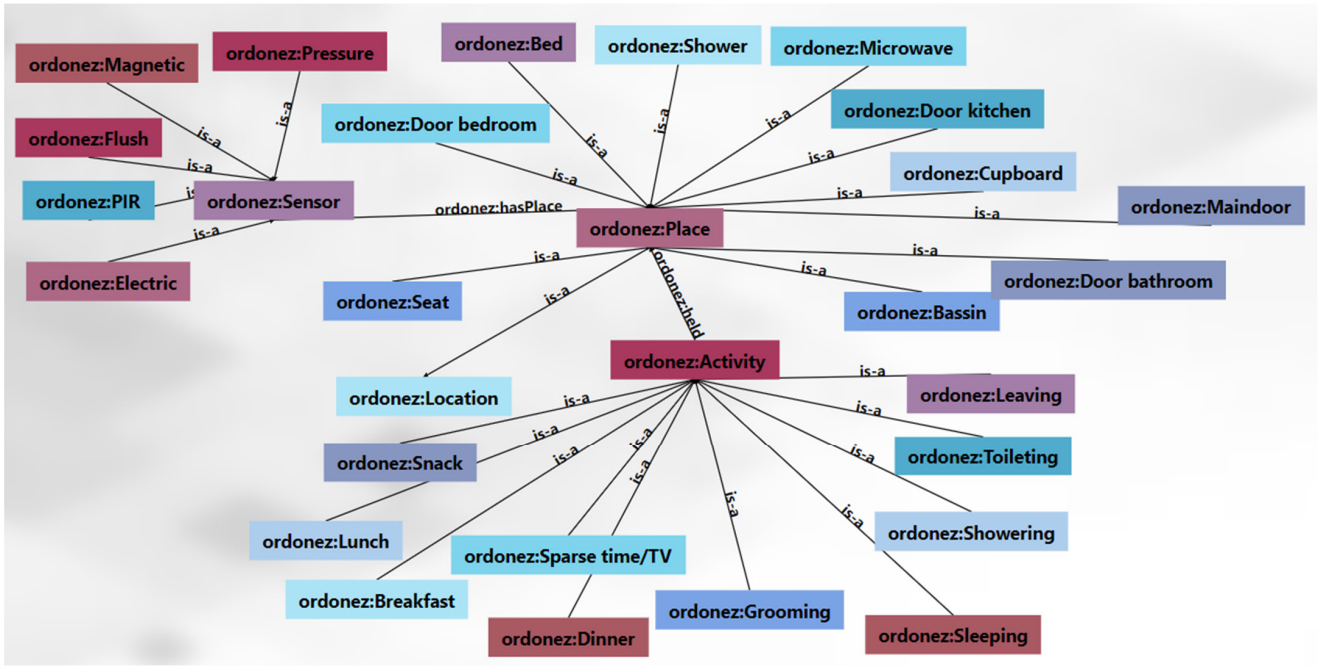


Figure 9. Ontology-based model learning at t_1, the refinement phase, showing an excerpt of the refined local ontology.

After the refinement, the last module updated the user's initial ontology-based model. As a result, at t_1, an evolved ontology-based model was generated, an excerpt of which is depicted in Figure 10.

- Scenario 2: Moving from a smart apartment to a smart home.

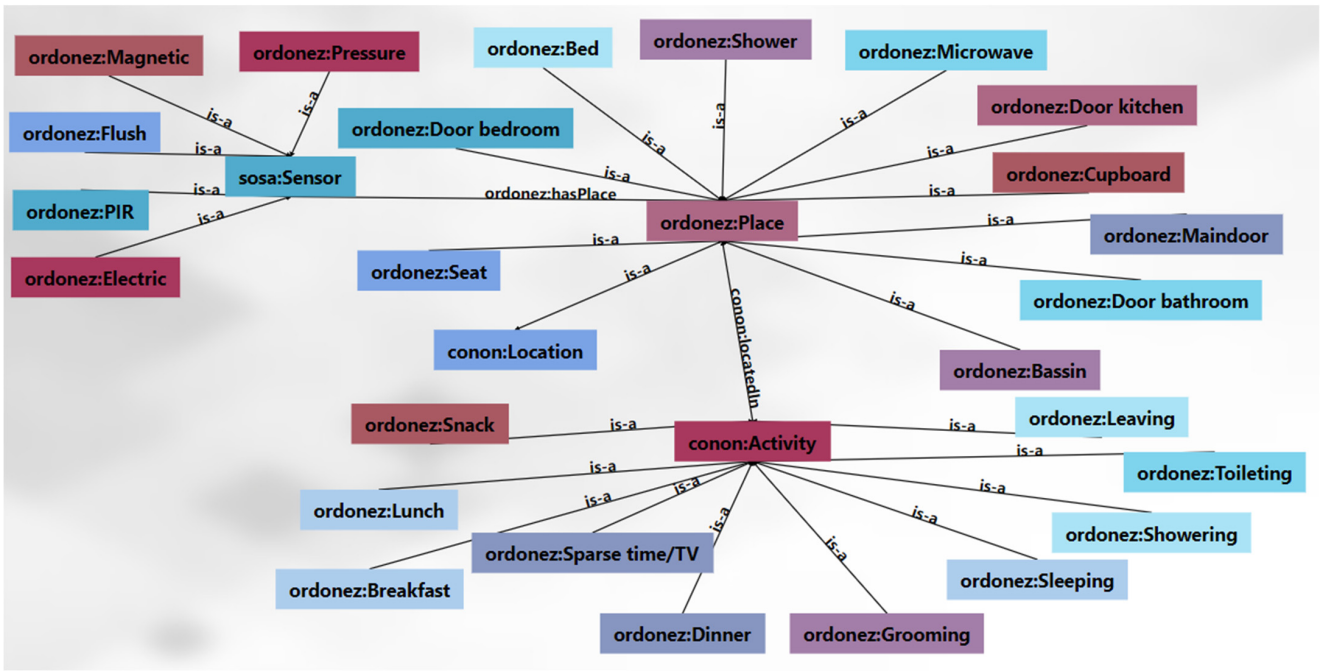
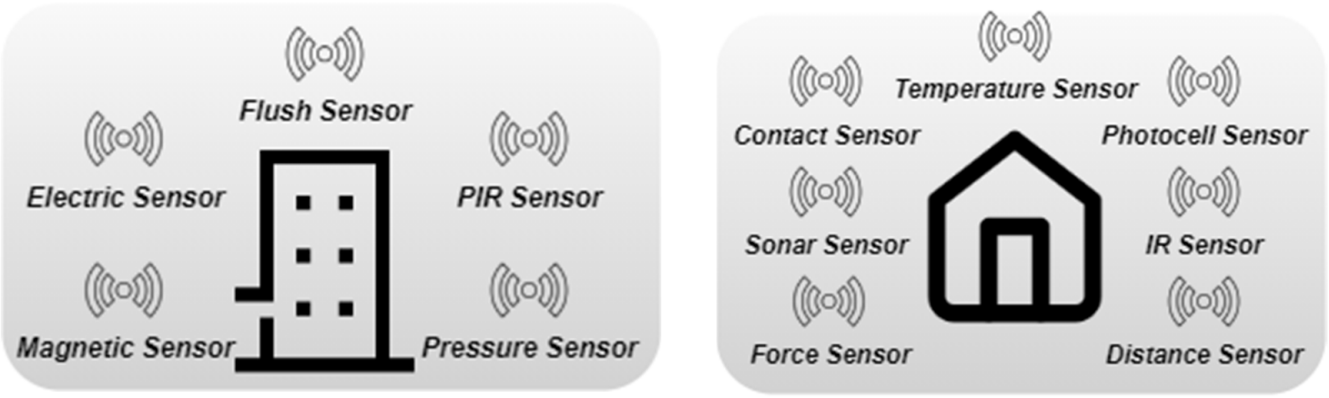


Figure 10. Ontology-based model evolution at t_1, showing an excerpt of the evolved ontology-based model at t_1.

In this scenario, the user lived in a smart apartment at t_1. This apartment encompassed a set of sensors, which is represented in Figure 11a. Obviously, the user's initial ontology-based model answered to his or her surrounding environment as depicted in Figure 10. Then, after a period of time, the user moved out of his or her smart apartment and into

an independent smart home with more advanced sensors. At t_2, this new and different pervasive environment, described in Figure 11b, was visited by the user. As illustrated in Figure 11b, a range of sensors including distance, sonar, force, temperature, photocell, and contact to infrared (IR) were available in the user's new smart home.



(**a**) (**b**)

Figure 11. User's surrounding environments (**a**) at t_1 and (**b**) at t_2.

Consequently, the initial ontology-based model, previously presented in Figure 10, would answer to the new pervasive environment illustrated in Figure 11b. To provide a strong answer, the data source selection service applied the automated data source search engine to select the appropriate data source using Apache Lucene. Figure 12 shows that it adopted the ARAS dataset [27] for the evolution of the user's model at t_2, since this dataset had a stronger tendency for having the same features of the new smart home. The ARAS dataset is well-known in the literature for having collected and published a home automation dataset for daily living in smart homes. It contains 5,184,000 observations in CSV format conducted with 20 sensors. These sensors captured about 27 different activities. The candidate dataset was accompanied by metadata, a sample fragment of which is depicted in Figure 13.

Next, the data source format unification service concerning the generation of a hierarchical XML pivot model from the candidate dataset was performed. It started by parsing the candidate ARAS dataset after performing the preprocessing and preparation step to extract its schema information as depicted in Figure 14. Then, the set of mapping rules previously described in Table 2 was explored to transform the parsed candidate dataset to XML data. The output was a hierarchical XML pivot model including an XML document together with an XML schema document, whose results are represented in Figure 15.

Then, the previously generated XML schema was traversed to parse its elements and attributes using the XSOM parser. The output of XSOM is an object model document representing the different retrieved elements and attributes with their values as outlined in Figure 16. For instance, the "SuperElem_Activity" feature was defined as a simple type, while the "Going Out" and "Having Shower" features were defined as a complex type with a complex content extension.

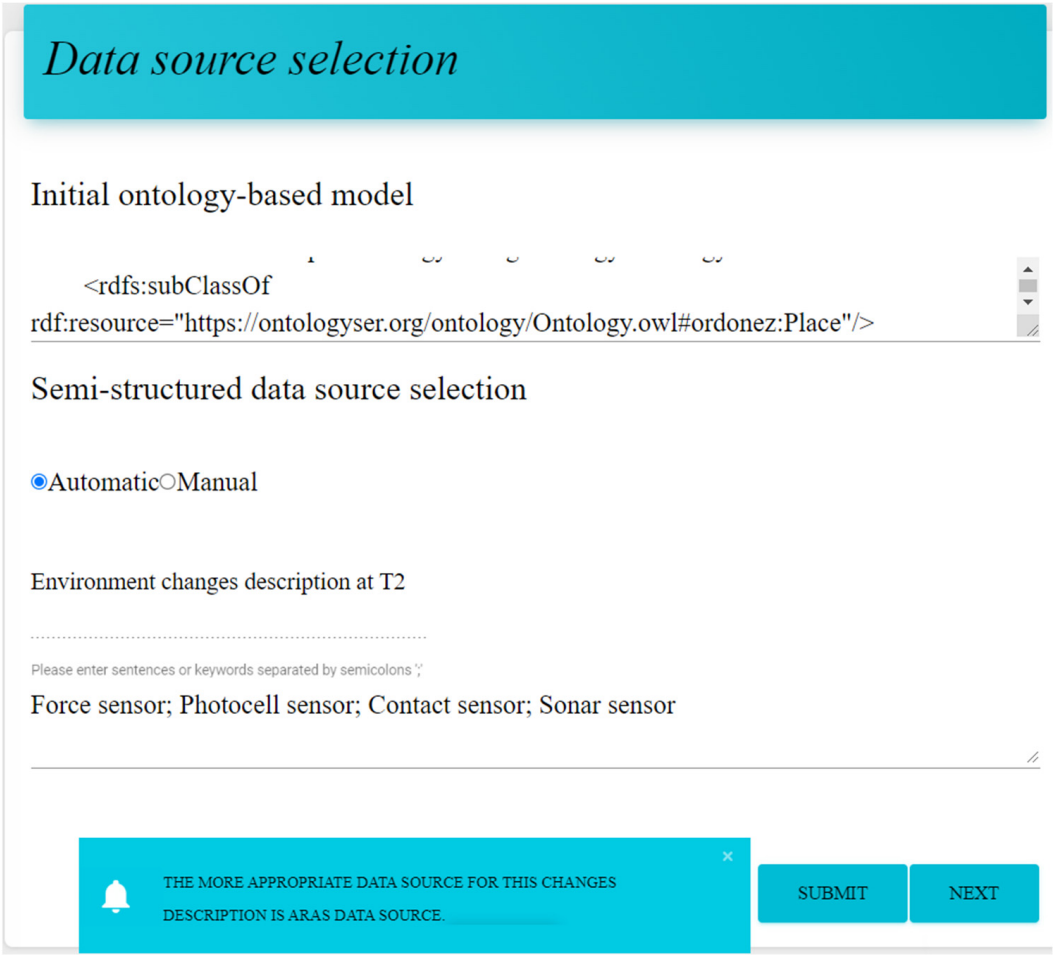


Figure 12. Candidate data source selection at t_2 of the ARAS dataset.

```
SENSOR EXPLANATION
There are several kinds of sensors: Photocell, Contact Sensor, Distance, Force
Sensor, Sonar Distance, Temperature and IR.

Each sensor has its place.

PLACE EXPLANATION
Places including Wardrobe, Couch, TV receiver, Chair, Fridge, Kitchen Drawer,
Bathroom Cabinet, House Door, Bathroom Door, Shower Cabinet Door, Hall, Kitchen,
Tap, Water Closet and Bed.

ACTIVITY EXPLANATION
Activity such as Changing Clothes, Having Guest, Having Conversation, Cleaning,
Listening to Music, Talking on the Phone, Brushing Teeth, Shaving, Laundry, Reading
Book, Using Internet, Napping, Toileting, Having Shower, Studying, Watching TV,
Sleeping, Having Snack, Washing Dishes, Having Dinner, Preparing Dinner, Having
Lunch, Preparing Lunch,Having Breakfast, Preparing Breakfast, Going Out, Other,
are included in ARAS dataset.

These activities performed at specific location such as home.
```

Figure 13. A sample fragment of the ARAS metadata.

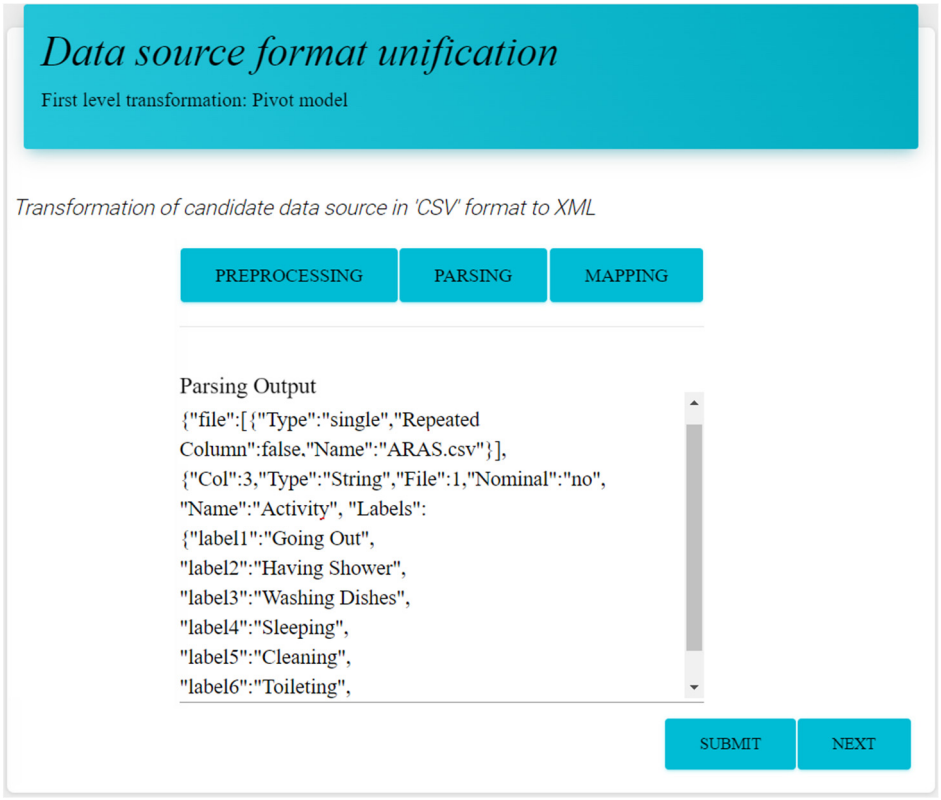


Figure 14. Data source format unification at t_2 with an example of a parsed candidate data source.

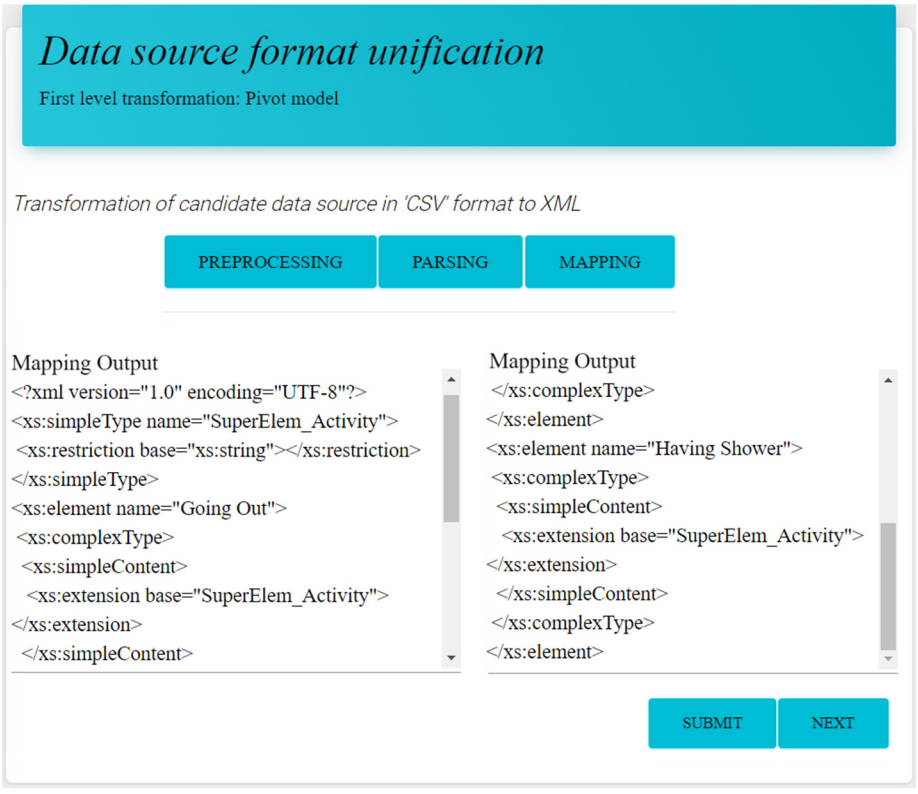


Figure 15. Data source format unification at t_2 with an example of the XML schema (XSD).

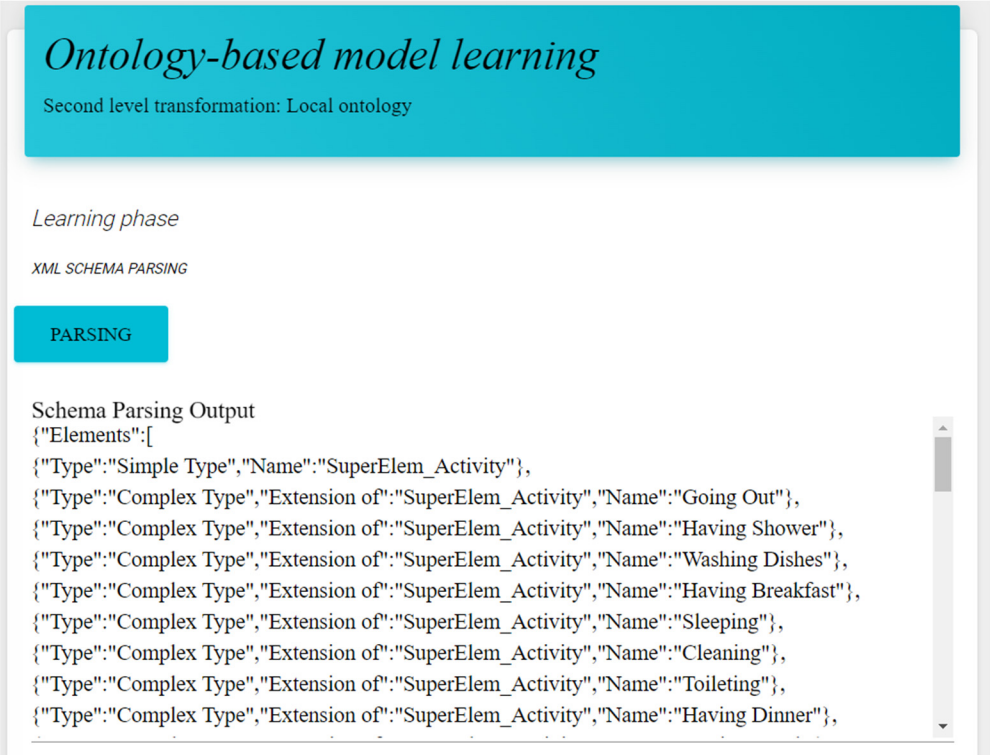


Figure 16. Ontology-based model learning at t_2 in the learning phase, an example of a parsed XML schema.

After that, the third module used this output as an input together with the set of mapping rules defined in Table 3 to generate a local ontology model using the Jena API. The second output of the third process was the resulting local ontology, where an excerpt of the local ontology is shown in Figure 17. Thus, for example, the two complex type elements "Going Out" and "Having Shower" were mapped to ontology sub-concepts, whose super concept was the "SuperElem_Activity" simple type element.

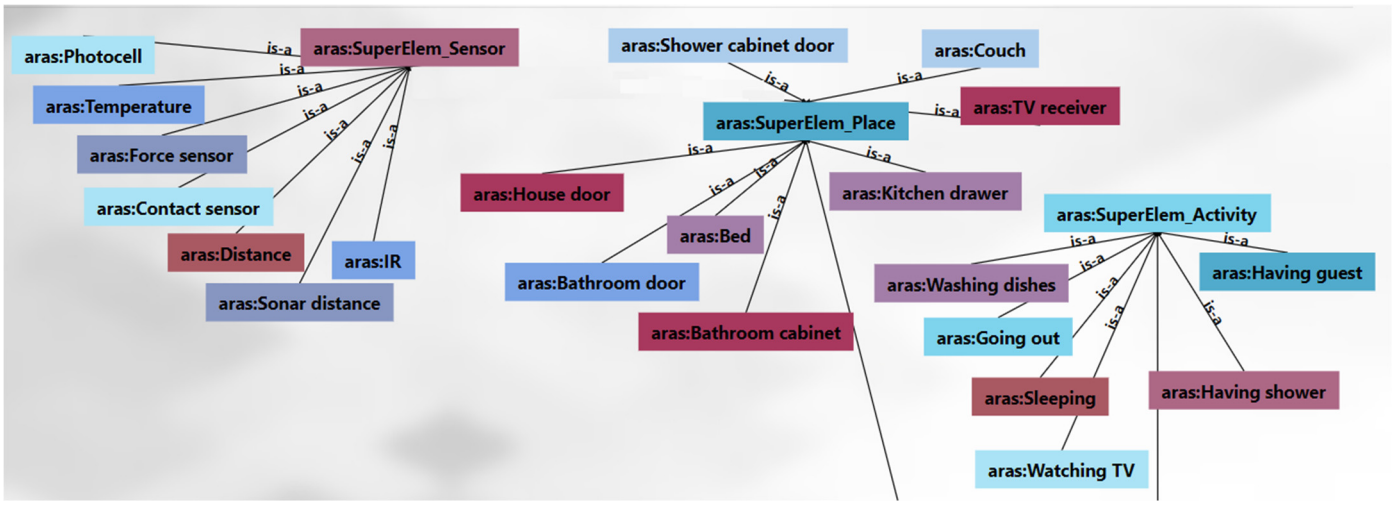


Figure 17. Ontology-based model learning at t_2 in the learning phase, an excerpt of the local ontology.

Subsequently and secondly, this module allowed for refining the local ontology and generating new elements if necessary. In this case, a taxonomic refinement was performed where, internally, the Hearst and Cea patterns and Matcher Java regex classes were used to recognize sentences that were included in the ARAS metadata and match them with such

patterns for identifying missing taxonomic relations as shown in Figure 18a. For instance, the second sentence matched well with Hearst's pattern "NP {,} such as {NP,} * {and | or} NP", and a new taxonomic relation was identified between the "Going Out" concept and "Activity" concept, since "Activity" is a hypernym of "Going Out". Consequently, the "aras:SuperElem_Activity" was replaced by the "aras:Activity" concept. Additionally, extJWNL was exploited to accomplish the taxonomic refinement as shown in Figure 18b. Thus, a new taxonomic relation was suggested between the "aras:Place" concept and "aras:Location" concept. Therefore, a definition of the "aras:Location" concept was created, and the suggested relation was built. Figure 19 presents an excerpt of the partially refined local ontology where all the taxonomic refinements are illustrated.

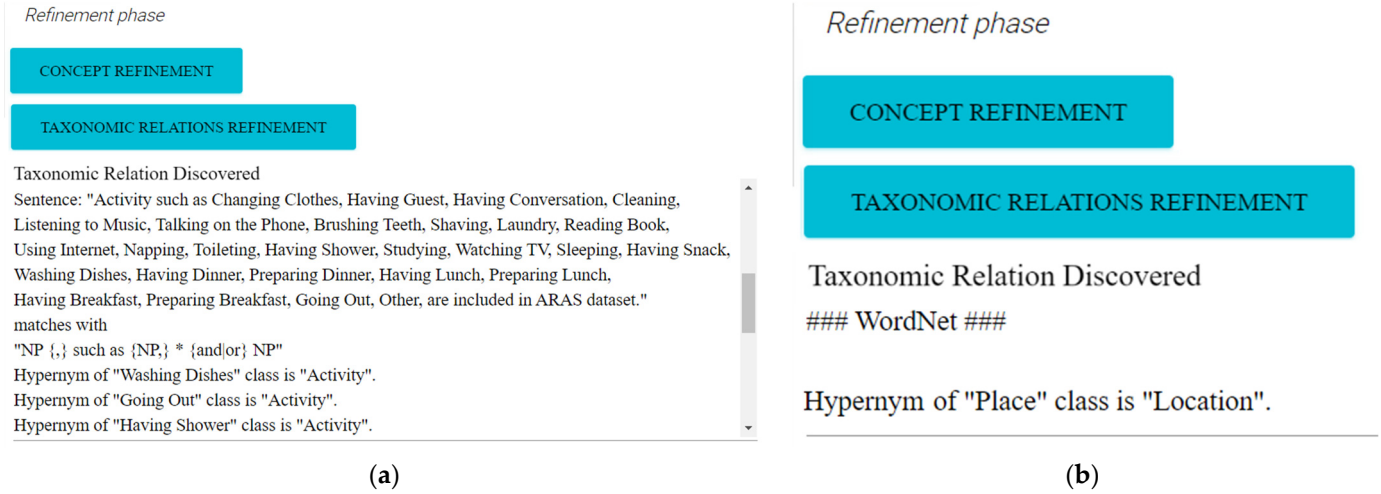


Figure 18. Ontology-based model learning at t_2 in the refinement phase. (**a**) An example of taxonomic refinement using LSPs. (**b**) An example of taxonomic refinement using WordNet.

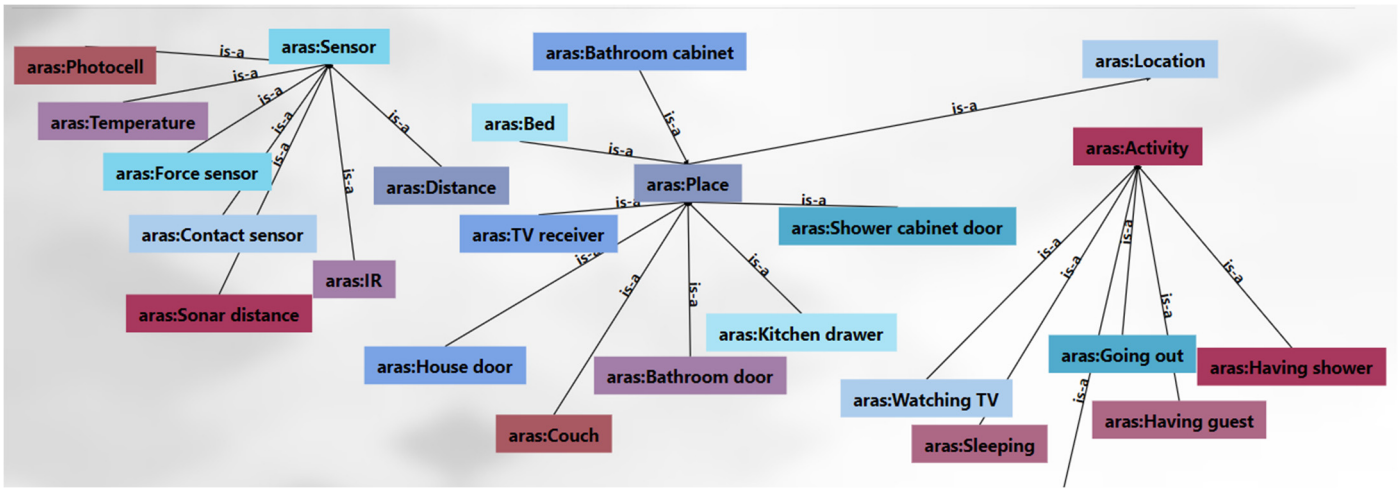


Figure 19. Ontology-based model learning at t_2 in the refinement phase, an excerpt of a partially refined local ontology after taxonomic refinements.

Afterward, non-taxonomic relation refinement was accomplished, since the hidden connections between all the learned super concepts were missing. In this case, for example, the metadata was looped to extract all the sentences that contained the label of the "aras:Sensor" concept. Then, analysis of the structure and dependencies of the candidate sentence was performed, and as illustrated in Figure 20, the <"Sensor", "has", "Place"> triplet was extracted, where the "has" verb was an indicator for a non-taxonomic relation and used to label the new non-taxonomic relation between the "aras:Sensor" and

"aras:Place" concepts. Figure 21 shows an excerpt of the refined local ontology, where all the non-taxonomic refinements are illustrated.

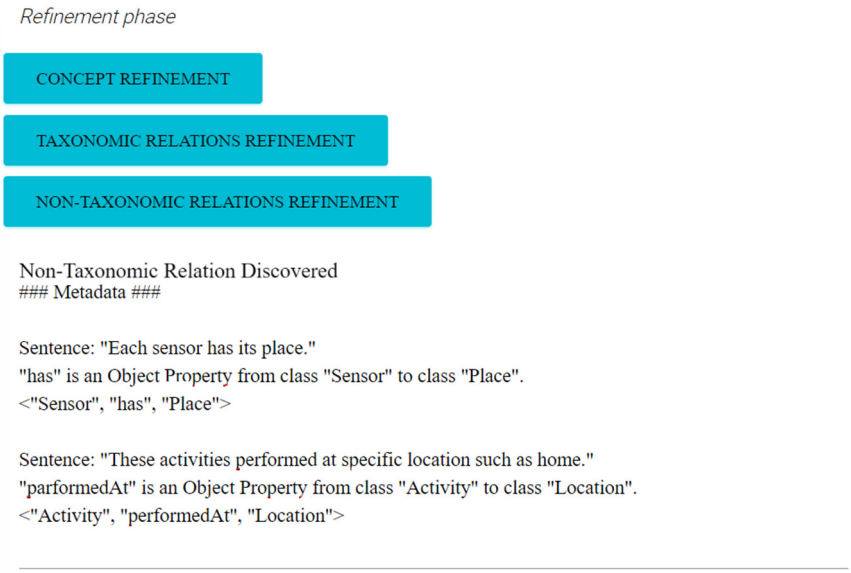


Figure 20. Ontology-based model learning at t_2 in the refinement phase, an example of non-taxonomic refinements.

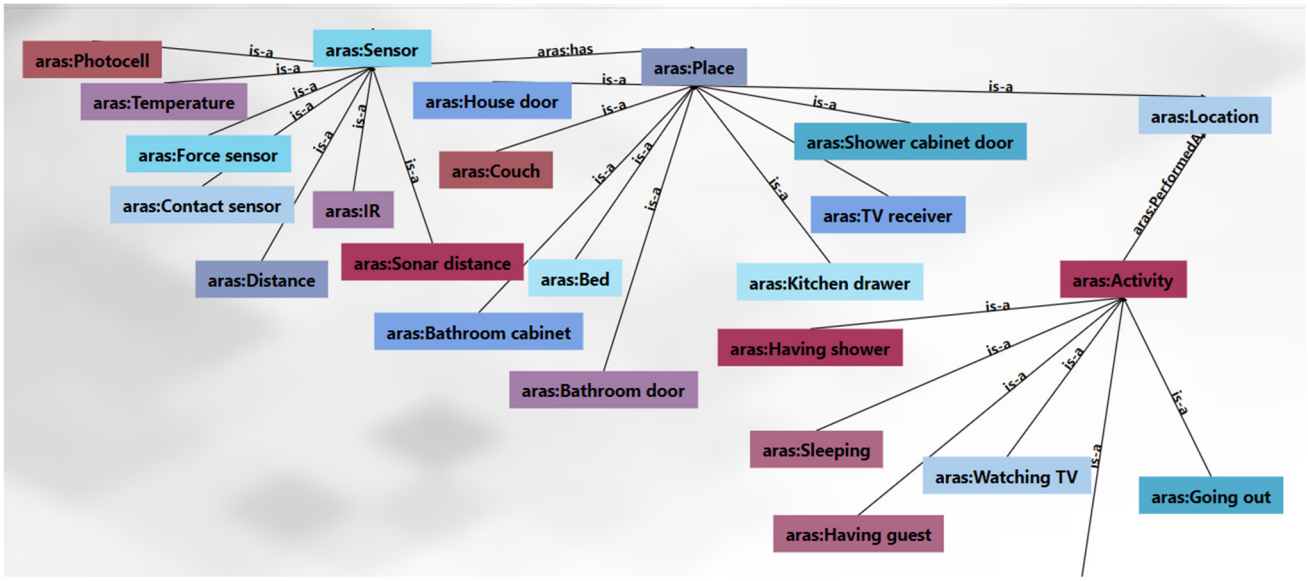


Figure 21. Ontology-based model learning at t_2 in the refinement phase, an excerpt of a partially refined local ontology after non-taxonomic refinements.

After the refinement phase, the last module supported the evolution of the user's initial ontology-based model. For this, the syntactic and semantic similarities were applied to find out the similarities between the ontology terms during the alignment activity. Figure 22 shows a part of the alignment findings. For example, the "aras:Location", "aras:Sensor", and "aras:Activity" concepts were equivalent to the "conon:Location", "sosa:Sensor", and "conon:Activity" concepts, respectively. Finally, the user's initial ontology-based model evolved based on the alignment results. The output of this process was the user's evolved ontology-based model at t_2, an excerpt of which is depicted in Figure 23. For instance, in the case of equivalent concepts with similar labels, such as "aras:Activity" and "conon:Activity", the original concept "conon:Activity" was kept, while in the case of dissimilar classes,

such as "ordonez:Pressure" and "aras:Temperature", the new concept "aras:Temperature" was copied, and the old one, "ordonez:Pressure", was overlooked. In addition, for the case of relations, the "aras:PerformedAt" relation was neglected, since there existed a relation "conon:locatedIn" that held between the concept "conon:Activity" and the concept "conon:Location". By contrast, the new relation "aras:hasPlace" was copied into the evolved ontology-based model to relate between the old concept "sosa:Sensor" and the new concept "aras:Place".

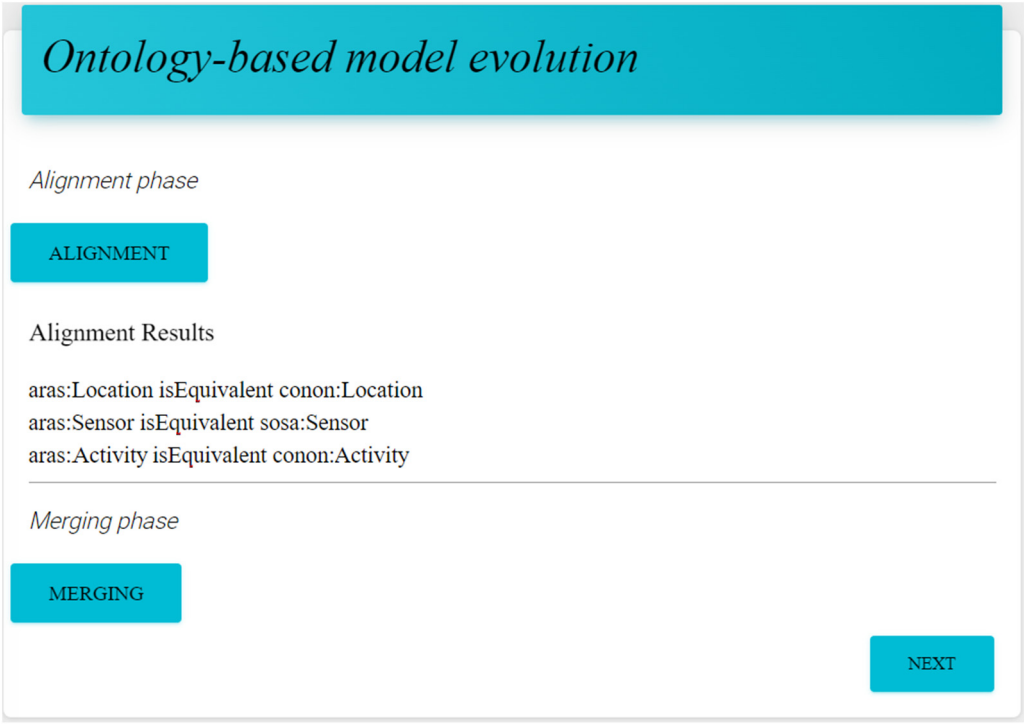


Figure 22. Ontology-based model evolution at t_2 in the alignment phase, an example of the main alignment findings.

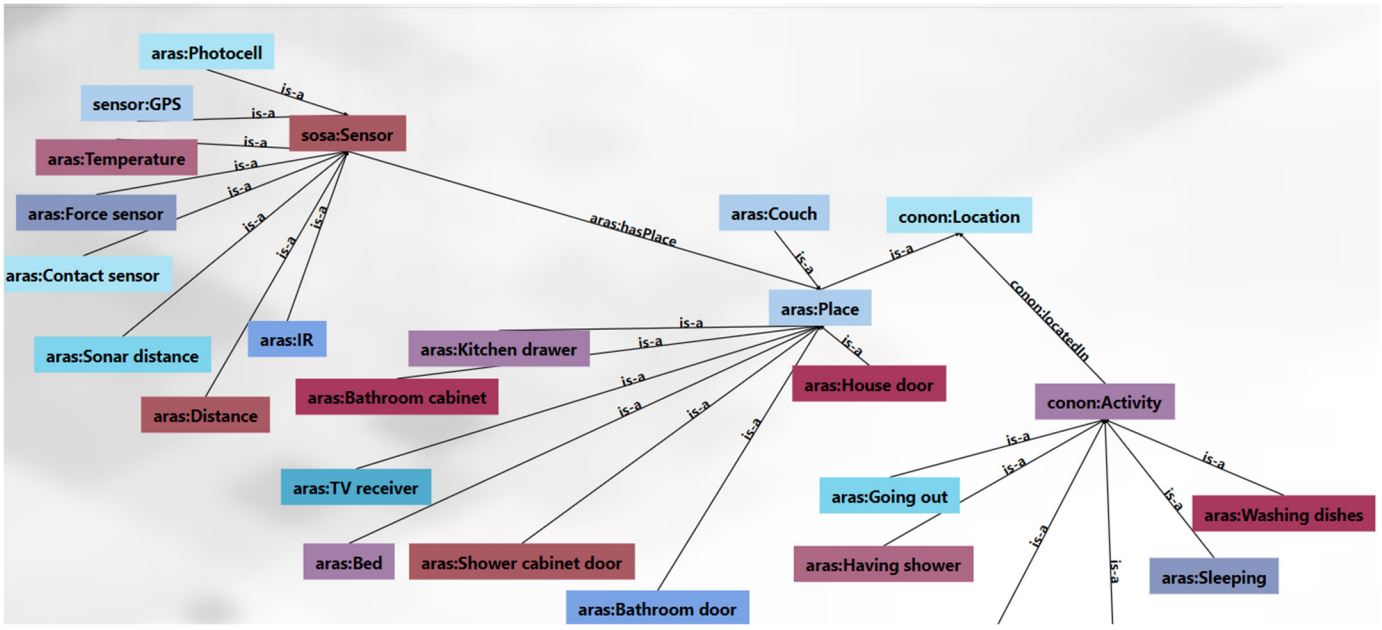


Figure 23. Ontology-based model evolution at t_2 in the merging phase, an excerpt of the evolved ontology-based model at t_2.

5. Evaluation

In this section, we aim to evaluate the quality of the evolved ontology-based models from different perspectives. For this purpose, the evaluation is threefold, where feature-based, criteria-based, and expert-based evaluation approaches are applied.

5.1. Feature-Based Evaluation

Feature-based evaluation is generally oriented toward evaluating the structural and schema aspects of ontologies to draw conclusions about their structure and schema quality. This evaluation has been adopted by different techniques, such as OntoClean, ONTOQA, and OntoMetrics. For our evolved ontology-based models, OntoMetrics [28] was used as a feature-based evaluation framework. It is a Web-based tool that consists of diverse metrics, such as base, schema, class, knowledge base, and graph metrics. In this work, we aimed to reuse two main metrics from OntoMetrics, called schema metrics and graph metrics, to determine the structure and schema quality with respect to the concepts, relations, and inheritance levels of the evolved ontology-based models. An overview of these metrics is discussed as follows:

- The schema metrics assess the design of ontologies by calculating and comparing statistics about the concepts, inheritance levels, relation types, properties, and other elements. They stand for searching for schema-related errors such as recursive definitions, unconnected concepts, missing domains or ranges, and missing inverse relations. The most essential and significant metrics of the schema category are the following: the Inheritance Richness (IR), Relationship Richness (RR) and Inverse Relations Ratio (IRR).
    - The IR is a way to measure the overall levels of the distribution of concepts of the ontology's inheritance tree. It is known as the average number of sub-concepts per concept to describe how the concepts are distributed across the different levels of the ontology and thus distinguish shallow from deep ontologies. A relatively low IR result would correspond to a deep or vertical ontology that covers its targeted environment in a very detailed way, while a high result, by contrast, would reflect a shallow or horizontal ontology that tends to represent a wide range of general concepts with fewer levels.
    - The RR examines the existing relations within an ontology to reflect the diversity of relationships. It is calculated as the fraction of the number of non-taxonomic relations, specifically the object properties, and the total number of sub-concepts and non-taxonomic relations in the ontology. The RR result is a number between 0 and 1, where a high value closer to 1 indicates that the ontology is rich and contains a variety of non-taxonomic relations, while a small RR value closer to 0 indicates that the ontology mostly consists of subsumption relations.
    - The IRR illustrates the ratio between the inverse non-taxonomic relations and all non-taxonomic relations. Lower values for this metric indicate a deficiency in the definition of inverse non-taxonomic relations in the ontology.
- Graph metrics are also known as structural metrics, where the taxonomy of ontologies is analyzed. These metrics calculate the cardinality and depth of the ontology structure in terms of the absolute and average depth, breadth, and so on. The depth metric that consists of an absolute and average is associated with the cardinality of the paths. The breadth metric, which is represented by the absolute and average, expresses the cardinality of the levels. The value of these different parameters in the graph metrics depicts the effectiveness of an ontology structure.

5.2. Criteria-Based Evaluation

The criteria-based evaluation measures the quality of the ontology content using common criteria that are identified in the ontology evaluation literature [29,30], such as consistency, conciseness, and so on. In our case, this evaluation was mainly concerned

with the content quality of an evolved ontology. Thus, this evaluation was carried out by evaluating the content of our evolved ontology-based models against particular criteria: (1) consistency, (2) completeness, and (3) conciseness. These criteria with their descriptions are presented in Table 6. We considered using the evaluation tool OOPS! [31], which stands for Ontology Pitfall Scanner, to check the consistency, completeness, and conciseness of the evolved models. OOPS! is a web-based evaluation tool for evaluating ontologies against a set of common design pitfalls. These potential pitfalls are classified into three levels of importance: critical, important, and minor. Minor pitfalls do not cause any severe problems, but their correction can improve the quality of the ontology. Important pitfalls refer to problems that are not critical to the consistency of the ontology but are considered important to correct. Critical pitfalls give rise to severe problems that may affect the consistency or reasoning of the ontology.

Table 6. Basic criteria for evolved ontology-based models' content evaluation.

Criteria	Description	Related OOPS! Pitfalls
Consistency	To ensure that the evolved ontology-based models do not contain any inconsistencies (e.g., contradictory or conflicting output results).	P05: Define incorrect inverse relationship. P06: Involve cycles in hierarchy. P07: Merging dissimilar concepts in the same concept. P19: Swapping intersection and union. P24: Using recursive definition.
Completeness	To ensure that all output results that are supposed to be in the evolved ontology-based models are explicitly presented.	P04: Creating unconnected ontology elements. P11: Missing domain or range in properties. P12: Missing equivalent properties. P13: Inverse relationships not explicitly declared.
Conciseness	To ensure that the evolved ontology-based models do not include redundancies (e.g., irrelevant or redundant output results).	P02: Creating class synonyms. P03: Creating "is" relationship place of "rdfs:subClassOf", "rdf:type", or "owl:sameAs". P21: Using a miscellaneous concept.

5.3. Expert-Based Evaluation

In an expert-based evaluation, the quality of the ontologies is judged on the basis of expert opinion. While this is a subjective evaluation approach, it is frequently considered a good validation process because it relies on the deep knowledge of external experts who can explore the quality of an ontology. In our case, expert-based evaluation is more about assessing the quality of initial and evolved ontology-based models in terms of coverage of the current surrounding environment. We took into account the coverage of concepts to capture the sufficiency of the ontology concepts for representing the runtime changes occurring in the enclosed environment. To this end, an expert was invited that had a good understanding of ontology development and engineering, such as ontology learning, alignment, and merging. He is considered an expert in his field since he has more than 10 years of experience. In this evaluation approach, the initial and evolved ontology-based models, together with a description of the surrounding environment changes, were given to the invited expert, who then applied his knowledge to assess the coverage of the given ontology-based models by checking all their concepts and identifying uncovered concepts to conclude their coverage in response to the emerging changes in the dynamic environment.

This evaluation consisted of calculating three well-known metrics: the precision, recall, and F-measure. The precision was used to indicate how accurately the concepts identified in an ontology-based model represented the current surrounding environment, and it was the number of correct concepts in the ontology-based model relative to the total number of concepts in the ontology-based model, as shown in Equation (2):

$$\text{Precision} = Nb\ of\ correct\ concepts\ in\ the\ model / Total\ Nb\ of\ concepts\ in\ the\ model, \quad (2)$$

The recall was used to measure the environment coverage of the ontology-based model, and it was the number of correct concepts relative to the total number of possible concepts, as shown in Equation (3):

$$\text{Recall} = Nb\ of\ correct\ concepts\ in\ the\ model / Total\ Nb\ of\ possible\ concepts, \quad (3)$$

The F-measure was used to measure the accuracy of the ontology-based model, and it was the harmonic mean that combined both the precision and recall values as shown in Equation (4):

$$\text{F-measure} = 2 \times (Precision \times Recall) / (Precision + Recall), \quad (4)$$

6. Results

This section presents the evaluation results obtained from feature-based, criteria-based, and expert-based evaluation approaches for evolved ontology-based models.

6.1. Evolved Ontology-Based Models' Overview

To perform the quality evaluation, we proposed assessing five ontology-based models evolved by our proof of concept. Table 7 shows the number of concepts, the number relations, and the information about the candidate datasets used for the evolution.

Table 7. Insights in the evolved ontology-based models.

	#Concepts	#Relations	Candidate Dataset			
			Name	Setting	#Dataset Size	#Dataset Observation
Evolved ontology-based model 1	68	80	ARAS [27]	Smart home	47	5,184,000
Evolved ontology-based model 2	37	54	Ordonez [26]	Smart apartment	24	20,358
Evolved ontology-based model 3	26	38	HHAR [32]	Ordinary apartment	16	43,930,250
Evolved ontology-based model 4	22	31	RCD [33]	Smart room	14	250,000
Evolved ontology-based model 5	139	167	ExtraSensory [34]	Outdoor or indoor	127	300,000

6.2. Feature-Based Evaluation Results

With regard to the evolved models, the mean results of the schema metrics are given in Figure 23. From Figure 24, it can be seen that we achieved a mean value for the IR equal to 0.903. This IR result proved that the evolved models were deep or vertical ontologies due to the fact that they offered several levels of inheritance, where each concept had at least two sub-concepts. Aside from this, the mean result of the RR was about 0.081. We can note that this result is close to zero, indicating that most relations defined in the evolved models were subsumption relations. It is obvious that the evolved models brought minimal non-taxonomic relations, which could be determined from the mean result of the RR. Furthermore, the obtained mean result of the IRR was equal to 0, which means there was a deficiency of inverse relations in the evolved models.

The mean results of the graph metrics are scattered in Figure 25. As this figure shows, the evolved models had a mean absolute depth of 135 and a mean average depth of 2.177. Thus, the depth metric results obtained can confirm the verticality of the evolved models as assessed by the previously obtained mean IR metric result. In turn, the evolved models presented a mean absolute breadth of 62 and a mean average breadth of 7.75. Thus, the breadth metric results reinforced the vertical hierarchical design of the evolved models.

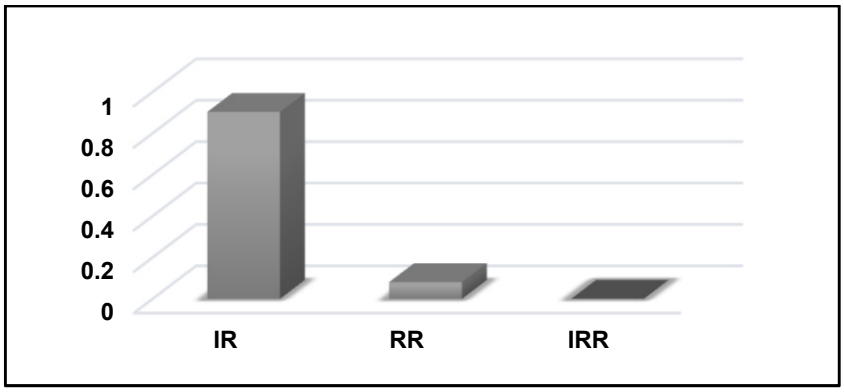


Figure 24. Mean results of schema metrics for the evolved ontology-based models using the OntoMetrics tool.

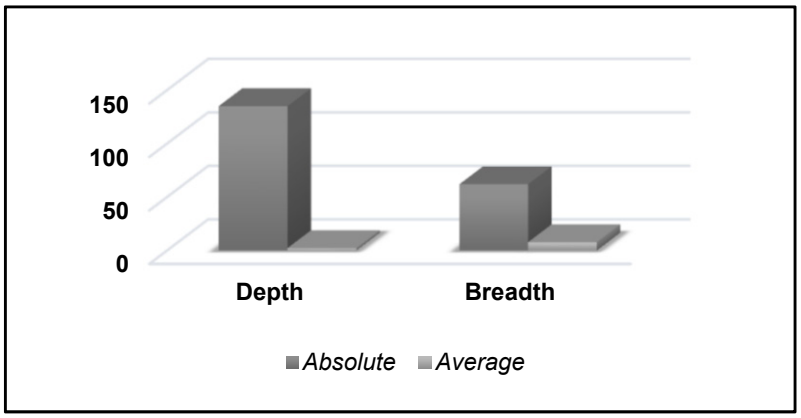


Figure 25. Mean results of graph metrics for the evolved ontology-based models using the OntoMetrics tool.

6.3. Criteria-Based Evaluation Results

A Web-based evaluation via OOPS! was performed, and its outcome is shown in Table 8. This table points out the different pitfalls that were encountered in the evolved models using OOPS! along with their specific descriptions. As reported in Table 8, the detected pitfalls did not affect the consistency or conciseness of the evolved models. On the contrary, OOPS! showed normal consistency and conciseness in the evolved models. Thus, the evolved models met both of these standards because they did not contain irrelevant or redundant output results and did not include any inconsistencies. By contrast, for completeness, a minor pitfall was returned regarding the inverse relationships not being explicitly declared. This pitfall was the evolved models omitting the declaration of inverse relationships. Thus, we can observe that the completeness pitfalls result correlated with the above-discussed mean result of the IRR to show that the evolved models were not complete. To fix the detected pitfall, the OOPS! guidelines proposed explicitly declaring inverse relationships in the evolved models.

Table 8. Pitfalls in evolved ontology-based models detected by OOPS!

Criteria	Detected Pitfall	Affects to	OOPS! Importance Level	Satisfaction
Consistency	-	-	Normal	Yes, no contradictory or conflicting output results can be inferred by reasoners since OOPS! shows no errors for all evolved models.
Completeness	P13: Inverse relationship not explicitly stated	Non-taxonomic relations	Minor	No, the evolved models are not completed well since inverse relationships were not explicitly defined as determined by OOPS!
Conciseness	-	-	Normal	Yes, no unnecessary or redundant output results were contained in the evolved models according to OOPS!

6.4. Expert-Based Evaluation Results

In the first round of the expert-based evaluation, the invited expert explored the changes occurring in the surrounding environment and then navigated across the initial and evolved ontology-based models. In the second round, he evaluated both models' quality in terms of environment coverage. Consequently, the precision, recall, and F-measure of the initial and evolved ontology-based models were computed as shown in Table 9.

Table 9. Initial and evolved ontology-based models' coverage results.

	Precision	Recall	F-Measure
Initial ontology-based model	0.69	0.48	0.57
Evolved ontology-based model	0.80	0.65	0.72

Considering the results presented in Table 9, it is clear that the evolved model achieved a precision score of 0.80 and a recall score of 0.65. In addition, we obtained a quite good result for the F-measure of 0.72, knowing that the F-measure's best value is at 1 and its worst value is at 0. As a whole, the evolved model's coverage results were promising and showed considerable precision, recall, and F-measure values. According to these results, we observe that the evolved model could fit well with the surrounding environment changes, since it showed a higher coverage level than the initial model.

7. Discussion

In this section, we discuss the main obtained results after completing the evaluation process. For simplifying the discussion of the results, they are explained in the following according to each applied evaluation approach.

From the feature-based evaluation perspective, the schema metrics' results, particularly for the RR and IRR, reflected a relatively minimal amount of non-taxonomic relations in the evolved ontology-based models due to incompleteness in terms of the non-taxonomic relations and their inverses. In addition, the IR result underlined a vertical hierarchy in the evolved models with a large number of inheritance levels. In accordance with the IR result, the graph metrics emphasized the verticality of the evolved ontology-based models and guaranteed the structure's effectiveness. At the end, the obtained feature-based evaluation results could ensure the structure and schema quality of the evolved ontology-based models.

From the criteria-based evaluation perspective, the results showed that evolved ontology-based models could assure good consistency and conciseness. Conversely, the requirement for completeness was not met as well, since the evolved ontology-based models did not explicitly represent inverse relationships. Despite the noted issue, the obtained criteria-based evaluation results could confirm appropriate content quality for the given evolved ontology-based models.

From the expert-based evaluation perspective, the results indicated that the evolved ontology-based model exceeded in terms of environment coverage, which was proven by the higher scores for precision and recall. The improvement in the recall score was higher than that for precision, which reveals that the evolved ontology-based model was better in coverage against the initial ontology-based model. Therefore, the coverage requirement of the evolved ontology-based model was considered acceptable, since the evolved ontology-based model had the advantage of producing high precision and good recall. Despite this, there was room for improvement regarding the coverage requirement through providing minor enrichments, since the score of the F-measure led to around 0.72. Therefore, the expert stressed, in an attempt to achieve a higher coverage, the need for slight enrichments of the evolved ontology-based model with additional knowledge to foster the expert's coverage agreement.

To conclude, the overall results were promising and largely showed an appropriate quality of content, schema, and structure in the evolved ontology-based models. They reflected a considerable consistency, conciseness, and coverage, whereas the completeness was not met that well. These findings highlight the quality of the evolved ontology-based models for answering changes arising in the surrounding environments at runtime and achieving the purpose of the ontology-based model evolution approach.

8. Conclusions

Regarding the main concerns associated with the dynamic environment changes over time, we have presented an automatic ontology-based model evolution approach that takes advantage of ontology learning to answer dynamic environments' changes at runtime. The chief aim of the presented approach is to analyze heterogeneous, semi-structured input data for learning an ontology that will be used to extend an initial ontology-based model at runtime. This approach led to obtaining a more relevant picture of an ontology-based model regarding the surrounding environment changes without expert contributions at runtime. Therefore, evolved ontology-based models were evaluated through three evaluation approaches under different parameterizations. First, we provided a feature-based evaluation to assess the design and structure of the evolved ontology-based models. Secondly, we conducted a criteria-based evaluation of the evolved ontology-based models in terms of content through analyzing the evolved ontology-based models' consistency, conciseness, and completeness. Third, we used an expert-based evaluation to assure the initial and evolved ontology-based models' coverage in the current enclosed environment. These evaluation findings reflected the quality of the evolved ontology-based models in supporting environments' changes at runtime. However, we found that the results were still insufficient, because we did not assess the relevance of the data sources according to the environment changes. In this sense, we plan to further improve the evaluation and assess the evolved ontology-based models from the point of view of data source relevance. Additionally, an advancement beyond the deficiency of inverse relationships among concepts is intended to be considered to enhance the presented approach.

Author Contributions: Conceptualization, R.J.; methodology, R.J.; software, R.J.; validation, R.J.; formal analysis, R.J.; investigation, R.J.; writing—original draft preparation, R.J.; writing—review and editing, R.J., M.K. and F.B.; visualization, S.F.; supervision, M.K. and F.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Echarte, F.; Astrain, J.J.; Córdoba, A.; Villadangos, J.E. Ontology of Folksonomy: A New Modelling Method. *SAAKM* **2007**, *289*, 36.
2. Rani, M.; Dhar, A.K.; Vyas, O. Semi-automatic terminology ontology learning based on topic modeling. *Eng. Appl. Artif. Intell.* **2017**, *63*, 108–125. [CrossRef]
3. Krataithong, P.; Buranarach, M.; Hongwarittorrn, N.; Supnithi, T. Semi-automatic framework for gener-ating RDF dataset from open data. In *International Symposium on Natural Language Processing*; Springer: Cham, Switzerland, 2016; pp. 3–14.
4. Yao, Y.; Liu, H.; Yi, J.; Chen, H.; Zhao, X.; Ma, X. An automatic semantic extraction method for web data interchange. In Proceedings of the 2014 6th International Conference on Computer Science and Information Technology (CSIT), Amman, Jordan, 26–27 March 2014; IEEE: Washington, DC, USA, 2014; pp. 148–152.
5. Booshehri, M.; Luksch, P. An Ontology Enrichment Approach by Using DBpedia. In Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, Larnaca, Cyprus, 13–15 July; ACM Press: New York, NY, USA, 2015; pp. 1–11.
6. Gómez-Pérez, A.; Manzano-Macho, D. A survey of ontology learning methods and techniques. *OntoWeb Deliv. D* **2003**, *1*.
7. Asim, M.N.; Wasim, M.; Khan, M.U.G.; Mahmood, W.; Abbasi, H.M. A survey of ontology learning techniques and applications. *Database* **2018**, *24*, 2018. [CrossRef] [PubMed]
8. Lehmann, J.; Voelker, J. An introduction to ontology learning. *Perspect. Ontol. Learn.* **2014**, *18*, 7–14.
9. Ma, C.; Molnár, B. Use of Ontology Learning in Information System Integration: A Literature Survey. In *Machine Learning and Knowledge Discovery in Databases*; Springer: Vienna, Austria, 2020; pp. 342–353.
10. Jablonski, S.; Lay, R.; Meiler, C.; Muller, S.; Hümmer, W. Data logistics as a means of integration in healthcare applications. In Proceedings of the 2005 ACM symposium on Applied computing—SAC' 05, Santa Fe, NM, USA, 13–17 March 2005; ACM Press: New York, NY, USA, 2005; p. 236.
11. Völker, J.; Niepert, M. Statistical Schema Induction. In Proceedings of the 8th Extended Semantic Web Conference, Heraklion, Crete, Greece, 29 May–2 June 2011; Springer: London, UK, 2011; pp. 124–138.
12. Bohring, H.; Auer, S. Mapping XML to OWL ontologies. In *Marktplatz Internet: Von e-Learning bis e-Payment, Leipziger Informatik-Tage (LIT 2005)*; Gesellschaft für Informatik e. V.: Berlin, Germany, 2015.
13. Lakzaei, B.; Shmasfard, M. Ontology learning from relational databases. *Inf. Sci.* **2021**, *577*, 280–297. [CrossRef]
14. Sbai, S.; Chabih, O.; Louhdi, M.R.C.; Behja, H.; Zemmouri, E.M.; Trousse, B. Using decision trees to learn ontology taxonomies from relational databases. In Proceedings of the 2020 6th IEEE Congress on Information Science and Technology (CiSt), Agadir-Essaouira, Morocco, 5–12 June 2021; IEEE: Washington, DC, USA, 2020; pp. 54–58.
15. Aggoune, A. Automatic ontology learning from heterogeneous relational databases: Application in alimenta-tion risks field. In Proceedings of the IFIP International Conference on Computational Intelligence and Its Applications, Oran, Algeria, 8–10 May 2018; Springer: Cham, Switzerland, 2018; pp. 199–210.
16. Sbissi, S.; Mahfoudh, M.; Gattoufi, S. A medical decision support system for cardiovacsular disease based on ontology learning. In Proceedings of the 2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA), Tunis, Tunisia, 6–8 February 2020; IEEE: Washington, DC, USA, 2020; pp. 1–9.
17. Shamsfard, M.; Barforoush, A.A. The state of the art in ontology learning: A framework for comparison. *Knowl. Eng. Rev.* **2003**, *18*, 293–316. [CrossRef]
18. Khadir, A.C.; Aliane, H.; Guessoum, A. Ontology learning: Grand tour and challenges. *Comput. Sci. Rev.* **2021**, *39*, 100339. [CrossRef]
19. de Cea, G.A.; Gomez-Perez, A.; Montiel-Ponsoda, E.; Suárez-Figueroa, M.C. Natural Language-Based Approach for Helping in the Reuse of Ontology Design Patterns. In Proceedings of the Computer Vision, Acitrezza, Catania, Italy, 29 September–3 October; Springer: New York, NY, USA, 2008; pp. 32–47.
20. Almuhareb, A. Attributes in Lexical Acquisition. Ph.D. Thesis, University of Essex, Essex, UK, 2006.
21. Sowa, J.F. *Knowledge Representation: Logical, Philosophical and Computational Foundations*; Brooks/Cole Publish-ing Co.: Pacific Grove, CA, USA, 1999.
22. Hearst, M.A. Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 14th Conference on Computational Linguistics, Nantes, France, 23–28 August 1992; Volume 2, pp. 539–545.
23. Hearst, M. WordNet: An Electronic Lexical Database and Some of Its Applications. Automated Discovery of WordNet Relations. 1998. Available online: https://direct.mit.edu/books/book/1928/WordNetAn-Electronic-Lexical-Database (accessed on 9 November 2021).
24. Cederberg, S.; Widdows, D. Using lsa and noun coordination information to improve the recall and precision of automatic hyponymy extraction. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, Stroudsburg, PA, USA, 31 May 2003; pp. 111–118.
25. Apache Lucene Core. Available online: https://lucene.apache.org/core (accessed on 2 October 2021).

26. Ordóñez, F.J.; De Toledo, P.; Sanchis, A. Activity Recognition Using Hybrid Generative/Discriminative Models on Home Environments Using Binary Sensors. *Sensors* **2013**, *13*, 5460–5477. [CrossRef] [PubMed]
27. Alemdar, H.; Ertan, H.; Incel, O.D.; Ersoy, C. ARAS human activity datasets in multiple homes with multiple residents. In Proceedings of the 2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops, Venice, Italy, 5–8 May 2013; IEEE: Washington, DC, USA, 2013; pp. 232–235.
28. Lantow, B. OntoMetrics: Putting Metrics into Use for Ontology Evaluation. In Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Porto, Portugal, 9–11 November 2016; SCITEPRESS-Science and Technology Publications: Setubal, Portugal, 2016; pp. 186–191.
29. Gómez-Pérez, A. *From Knowledge Based Systems to Knowledge Sharing Technology: Evaluation and Assessment*; Knowledge Systems Lab., Stanford University: Stanford, CA, USA, 1994.
30. Guarino, N.; Welty, C. Evaluating ontological decisions with OntoClean. *Commun. ACM* **2002**, *45*, 61–65. [CrossRef]
31. Poveda-Villalón, M.; Gómez-Pérez, A.; Suárez-Figueroa, M.C. Oops! (ontology pitfall scanner!): An on-line tool for ontology evaluation. *Int. J. Semant. Web Inf. Syst. (IJSWIS)* **2014**, *10*, 7–34. [CrossRef]
32. Stisen, A.; Blunck, H.; Bhattacharya, S.; Prentow, T.S.; Kjærgaard, M.B.; Dey, A.; Jensen, M.M. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In Proceedings of the 13th ACM conference on embedded networked sensor systems, Seoul, Korea, 1–4 November 2015; pp. 127–140.
33. Morgner, P.; Müller, C.; Ring, M.; Eskofier, B.; Riess, C.; Armknecht, F.; Benenson, Z. Privacy implica-tions of room climate data. In *European Symposium on Research in Computer Security*; Springer: Cham, Switzerland, 2017; pp. 324–343.
34. Vaizman, Y.; Ellis, K.; Lanckriet, G. Recognizing Detailed Human Context In-the-Wild from Smartphones and Smart-watches. *arXiv* **2017**, arXiv:1609.06354.