An Upper-Ontology-Based Approach for Automatic Construction of IOT Ontology

Yuan Xu, Chunhong Zhang, and Yang Ji

Mobile Life and New Media Laboratory, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Yuan Xu; xyuanu2011@gmail.com

Received 7 November 2013; Accepted 4 February 2014; Published 3 April 2014

Copyright © 2014 Yuan Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ontology gives us a reliable group of concepts and the relations between concepts in an IOT system. It does not only save words of format but also accurately transfers semantic data between human users and the computers. Hence, the usefulness of resources in IOT system depends on whether the domain ontology can be constructed effectively and correctly. In this paper we propose an automated method to construct the IOT ontology. First, we explain the necessity of introducing ontology automatic construction in IOT system and summarize the major challenges in existing approaches. Secondly, we introduced the existing ontology construction methods and summarize their issues. Thirdly, we give a framework of our ontology construction and research the key algorithms in detail: (1) knowledge-tuple extraction algorithm which contains contextual information; (2) concept semantic similarity algorithm which is based on the structure of tuple; (3) knowledge-tuple extraction model which is based on the structured information. Then we build a prototype and evaluate the ontology. Finally, we make conclusions and suggest directions for future research.

1. Introduction

Ontology is a normalized knowledge base as a set of concepts within a domain and the relationships between pairs of concepts [1]. Ontologies provide the structural frameworks for organizing information and are used in artificial intelligence, semantic web, systems engineering, software engineering, biomedical informatics, library science, and enterprise bookmarking. In Internet Of Thing (IOT) domain, experts extracted concepts from system architecture to construct the machine-understandable ontology and now there are more than ten IOT domain ontologies. Through the normalized representation for domain concepts and relationships, ontology provides a support service for semantic-based heterogeneous resource search and service development in IOT.

The current IOT domain ontologies are mostly based on artificial construct [2] by domain experts; the authority one is SSN ontology which has been built by W3C Semantic Sensor Network Incubator; in addition there are more than ten IOT domain ontologies include CSIRO, MMI, CESN, A3ME, and OntoSensor [3, 4]. These ontologies’ concepts are derived from the general knowledge of IOT domain such as “platform,” “gateway,” “device,” and “sensor;” they do not combine any actual IOT system and only contain a small set of concepts (tens to hundreds of concepts). At the same time, these ontologies have a slow improvement. But IOT system needs an ontology which combined system's features and provided a rich set of concepts (tens of thousands of concepts); in addition the ontology needs to improve itself with the development of IOT system constantly. Therefore, the existing IOT domain ontologies are not suitable for using in IOT systems directly.

There are many researches on ontology automatic construction in recent years, include TextOntoEx [5], TF-IDF-ART [6], and GRAONTO [7]. But these researches are still in primary stage. The automatic construction models typically include the following sections. In order to obtain the domain knowledge, constructors need to collect concerned texts and web pages as information source first. Then they use the word segmentation and remove stop words to wipe off the irrelevant vocabularies from texts and web pages. The knowledge tuple (which consists of concepts and relationships between concepts, such as “platform-has-gateway” and “gateway-has-device”) is the basic unit of ontology and an ontology is combined by a lot of knowledge
tuples; therefore, constructors need to extract knowledge tuples (mostly they extract the subject-predicate-object triple tuples from sentences as the knowledge tuples, subject and object are concepts, predicate is relationship). Taking into account the knowledge tuples are independent in existence, the constructors need to calculate the similarity between concepts in knowledge tuples and merge the similar concepts and constitute a connected ontology.

Existing tuple extraction mainly use unstructured documents [5–7] (refer to information that either does not have a predefined data model or is not organized in a predefined manner, such as text and web page) as information source, and commonly uses the natural language processing algorithms to extract knowledge tuples from each sentence of the information’s source; this approach makes each tuple lose context information in paragraph. In addition, IOT environment includes various resource description files (a type of structured information source, refers to information with embedded coding, such as markup, is used to give the whole and parts of the document with various structural meanings according to a schema) and this helps obtain a more comprehensive knowledge, but the existing models lack research on resource-description-based automatic construction. In the research on concepts’ similarity, existing approaches use linguistic similarity of vocabularies to merge concepts; this neglects the scene that the same word in different contexts has different meanings and represents different concepts. Moreover, [8] put forward that selecting an existing domain ontology as the upper ontology is the first step in classic five steps for ontology construction, but existing approaches mostly skip this step. We list the four major challenges and problems in this paper as follows.

(1) Upper ontology (also known as a top-level ontology or foundation ontology) is an ontology which describes very general concepts that are the same across one domain; it mostly is used to support a broad semantic interoperability between a large number of ontologies in a domain [9]. Selecting an existing domain ontology as the upper ontology can make full use of the most general concepts in this domain and support the interoperability with other ontologies, but existing construction models mostly neglect the interoperability between ontologies and skip this step. Therefore we will research the IOT upper-ontology evaluation and selection in Section 3.4.

(2) Knowledge tuples use concepts and relationships to describe the domain knowledge (e.g., platform-has-gateway, “platform” and “gateway” are concepts; “has” is relationship); they are the basic unit in ontology construction. Existing construction models commonly use the natural language processing algorithms to extract knowledge tuples from each sentence of information source; this approach neglects the context information in paragraph when extracting the tuples (e.g., we have a sentence “Common data formats include xml and json.” in an IOT platform description text, we can extract tuples “format-include-xml” and “format-include-json.” But without the context information, we will never know whether these data formats are used in IOT platform). Based on the discussion above, existing tuple extraction approaches lost the context information in paragraph. Therefore, we propose a context-information-based (CIB) knowledge-tuple extraction algorithm in Section 3.2.

(3) Existing knowledge-tuple extraction algorithms mostly rely on unstructured documents and neglect the structured documents. But IOT environment includes various resource descriptions, which are a type of structured documents. Resource description has been used to describe various IOT resources such as device and data. We use the Xively (previously Pachube, one of the most famous IOT platform) as an example; it has more than twenty thousand resource descriptions and exceeds the number of characters in other pages in this website. This helps to obtain a more comprehensive knowledge. Therefore we propose a resource-description-based (RDB) knowledge-tuple extraction model in Section 3.2.

(4) Existing concept similarity algorithms mostly use the linguistic similarity of vocabularies to merge synonyms, this neglects the scene that the same word in different contexts has different meanings and represents different concepts. This issue will lead to the concepts in ontology only that reflect the linguistic characteristics of words without any context information in specific scene (we use a couple of tuples as example, device-has-name-is-sensor and location-has-name-is-china; they have the common concept “name,” but they represent different context information and we cannot merge two “name” in ontology. If we merge them, we can get that china is also a name of device and it is obviously wrong). In linguistics, polyseme is a word or phrase with different senses. Majority of polyseme is verb and adjectival; they express one meaning in a certain context. According to statistics, each word has more than three meanings on average, so polysemy is a common phenomenon in human language. In IOT environment, there are inheritance relationships between concepts. Because of inheriting different characters from parent concepts, the vocabularies with same meaning in linguistics should reflect different concepts in ontology. Taking into account that the context information of a word in a tuple comes from the tuple structure, we propose a tuple-structure-based (TSB) similarity algorithm in Section 3.5.

The remainder of this paper is organized as follows. Section 2 introduces the related work on ontology automatic construction. The details of the proposed ontology construction method are explained in Section 3. The prototype and evaluation are shown in Section 4. Finally, conclusions and future work are shown in Section 5.
2. Related Work

Due to the critical importance of ontology in the various fields and applications, constructing domain ontology efficiently and effectively has been an important research area recently. Thus, different mechanisms and methodologies for designing and building ontology have been proposed. At present, most of the researches on ontology automatic construction are in accordance with the four steps in Section 1: (1) preprocessing the information sources; (2) extracting knowledge tuples from information sources; (3) calculating the semantic similarity between concepts and relationships in knowledge tuples, merging the similar concepts and relationships, constituting an initialization ontology; (4) calculating the importance and Boolean relations between concepts in initialization ontology, and constituting a hierarchical ontology.

In the research of extracting knowledge tuples, existing construction models lack an in-depth study for combining knowledge tuples with context information; this issue will affect the accuracy of tuple extraction directly. References [5, 10–13] put forward their own knowledge-tuple extraction method, respectively. Reference [10] extracts adjacent vocabularies as binary knowledge tuples in the text; this approach ignores the potential semantic relationships between nonadjacent vocabularies in papers. Reference [11] extracts knowledge tuples which are less than triple in the sentence based on pattern extraction algorithm. TextOntoEx [5] defines four modes of semantic pattern and divides the basic elements in semantic pattern into four categories: class, verbs, constants, and modifiers; then extracts knowledge tuples by pattern matching. Reference [12] proposed that the existing tuple extraction methods are generally based on shallow natural language process (NLP). The shallow NLP which is based on regular expressions will lose a lot of information in extraction; therefore, they use the Stanford depth NLP which is based on statistical parsing to extract knowledge tuples. References [5, 11, 12] all used the pattern extraction method which is based on NLP, but the length of the pattern analysis is confined to a single sentence and ignores the impact from context of the paragraph. Reference [13] proposed that the existing patterns are extracted in a single sentence, ignoring the relationship between sentence and paragraph. First, they use the paper’s title and other structured information to build a kernel ontology with hierarchical relationship, then supplement the ontology with unstructured information and get the integrity ontology; this approach considers the context information in tuple extraction preliminary. But this method is limited by the accuracy of the papers’ titles and processing the sentence regardless of the structured information still needs more in-depth search.

In the research of calculating the semantic similarity between concepts in ontology construction, existing construction models lack an in-depth study for structural similarity; this issue will lead to the concepts of ontology to only reflect the linguistic characteristics of words rather than the semantic features in the specific scene. References [11, 14–17] put forward their own semantic similarity measure. Reference [11] use the Chinese information source; they calculate the semantic similarity between concepts based on the Chinese semantic similarity tree from CKIP [18]. References [15, 16] measure the semantic similarity between concepts according to the words’ distance in WordNet dictionary. Reference [16] iterative improve the existing ontology through ontology merging technique; the relationship between concepts in initialization ontology and extended ontology also can be obtained by WordNet dictionary. The literature [11, 14–16] all used the linguistic similarity of individual word to measure the semantic similarity between words, ignoring the affect from knowledge tuples’ structure. Reference [17] introduced a set-similarity comparison in concepts similarity calculation. It achieved the adjacent concepts of objective concept and constituted an unordered set to measure the semantic similarity between concepts. This method considers the structural similarity in the condition of the adjacent nodes, but do not consider the similarity comparison in the view of mathematical model and still need more in-depth search.

The above references do not consider the introduction of existing domain upper ontology into automatic construction; also they do not considered the research for structured-information-based knowledge-tuple extraction model. Therefore, we will research the IOT upper-ontology evaluation and selection in this paper. At the same time, we propose a context-information-based knowledge-tuple extraction algorithm, a tuple-structure-based semantic similarity algorithm and a structured-information-based knowledge-tuple extraction model.

3. Upper-Ontology-Based Ontology Construction Model

3.1. Architecture of Ontology Construction System. In this section, we provide an overview of our architecture of ontology automatic construction. We describe the construction modules and data flows among different modules in Figure 1. We first research the construction models; ontology construction can be divided into two modes: global construction and incremental construction. In global construction, it needs to use all existing domain knowledge to rebuild the ontology when adding or changing knowledge. Global construction applies to a fully developed field; the knowledge of the field is complete and relatively static, without frequently adding or changing new knowledge. In incremental construction, when adding new knowledge, construction module will supplement and repair relevant parts in existing ontology according to the new knowledge. Incremental construction applies to an emerging field and the knowledge of the field is not complete. The field will put forward new knowledge frequently. Thus it requires a frequent iteration and updating of existing ontology. Compared with the global construction, incremental construction does not need to rebuild the ontology with all existing knowledge when adding new knowledge into the field; it only needs to update the relevant part with the new knowledge in ontology. Because the IOT domain has a mass of knowledge and is growing rapidly, incremental
construction will bring the advantages in computing and timeliness.

Our ontology construction model is an incremental construction model. We iterate the ontology through adding new knowledge tuple to existing ontology constantly. In order to guarantee the connectivity of ontology and reduce redundancy concepts, we need to calculate the similarity between concepts in new tuple and existing ontology when adding a new tuple. Through merging similar concepts, we will achieve the connection between new tuple and existing ontology. The architecture contains four modules. In order to obtain the normalized domain knowledge from information source (IS), we use the tuple extraction module to extract tuples from unstructured and structured files. Then normal module will use word stem and synonym to normalize the vocabularies in tuples. In order to make full use of the most general concepts in IOT domain and support the interoperability with other ontologies, we use upper ontology module to evaluate existing IOT domain ontologies and select SSN as our upper ontology. Then taking into account that the tuples are independent in their existence, we use the similarity module to calculate the structure similarity between concepts in new tuple and existing ontology. Then we merge the similar concepts and constitute a connected IOT ontology.

(1) In the tuple extraction module, we extract knowledge tuples with context information from unstructured IS and resource description files. Unstructured IS is a conventional form to represent the knowledge in a domain, such as thesis and related web pages. Therefore, we can use the natural language processing (NLP) algorithms to extract tuples and get the normalized knowledge in unstructured IS. Existing ontology automatic construction models extract knowledge tuples from each sentence of unstructured IS. This approach neglects the context information in paragraph and paper. Therefore, we propose the CIB tuple extraction algorithm to extract tuples with context information from unstructured files. In addition to extract triple tuples from each sentence based on NLP algorithms, we will extract the core concepts of paragraphs and papers based on the largest contiguous frequency statistics. Then we make the core concepts as the context labels and add them into triple tuples hierarchically. For example, we extract triple tuples (a)- (b)- (c); in addition the core concepts of this paper and paragraph are A and B; we can combine them and get the thorough tuple (A_B_a)- (b)- (A_B_c) (each concept in tuple is reflected with an order vector); this approach will retain the context information with concept prefix. Resource description file is another conventional information carrier in IOT domain; it is used to describe various IOT resources such as device and data. In order to get more comprehensive knowledge, we propose the RDB tuple extraction model to extract tuples from resource description files. In this model, we make use of the resource description which conforms to the tree structure and adopts the key-value model, then formulates three rules in Section 3.2.2 to extract tuples.

(2) In the normal module, we use the stem algorithm and WordNet dictionary to extract the concepts’ word stem and synonyms in tuples and we achieve the standardized representation for vocabularies. Because a vocabulary can have a diversified statement, such as word activity, abbreviation, and synonym (e.g., measure is the stem for measuring, IOT is the abbreviation for internet of things, and evaluation is the synonym of measure), thus, the same concept will have variety of representations in tuples. This will cause the redundancy in ontology; therefore, we need to normalize the vocabularies in tuples. Stemming of the words is to remove affixes and reduce inflected (or sometimes derived) words to their stem. Synonym is a unified process for a vocabulary set which has the same meaning in linguistics; we use one word to represent other words in a vocabulary set (e.g., measure, quantity, amount, evaluate, standard constitutes a synonyms set; we will use measure to represent other words in tuples). This operation will
provide a unique and standardized form for each vocabulary in tuples.

(3) In the upper-ontology module, we investigate the existing IOT domain ontologies and select SSN as our upper ontology. In order to make full use of the most general concepts in IOT domain and support the interoperability with other ontologies, we need to select IOT domain ontologies as our upper ontology. But existing automatic construction models mostly neglect the interoperability between ontologies and skip this step. We collect more than ten IOT domain ontologies such as SSN, CSIRO, MMI, and CESN. Through evaluating them in five aspects (key concepts, author, status, complexity, and cited), we choose the SSN as our upper ontology.

(4) In the similarity module, we calculate the structure similarity between concepts in tuples and merge the similar concepts to constitute a connected IOT ontology. Because the knowledge tuples in tuple extraction module are discrete, thus, the tuples cannot constitute a knowledge network and cannot be used for human or machines to understand the relevant knowledge of a concept. Existing approaches use the linguistic similarity of vocabularies to merge synonyms and neglect the scene that the same word in different contexts has different meanings and represents different concepts. Take into account that we add the multilevel context labs into tuples in tuple extraction module and each concept in tuples is expressed with a vector. Thus the structure of vector will reflect the context information of concepts in tuples; we propose the TSB similarity algorithm to calculate the structure similarity between concepts. Our ontology construction model is an incremental construction model. We iterate the ontology through adding new knowledge tuple to existing ontology constantly. Thus we need to extract contrast vector from ontology; then we need to make the concept in new tuple as object vector and use the longest common subsequence (LCS) between vectors to get structural similarity part between two vectors. Through calculating information content (IC) in structural similarity part, we can quantify the structural similarity between two vectors and provide evidence for merging similar concepts. Through merging the similar concepts, we can iterate the IOT ontology constantly.

Therefore, a closed loop for ontology construction is formed in the proposed approach. We will expound each module and related algorithm in subsequent sections.

3.2. Knowledge-Tuple Extraction Algorithm. Knowledge tuples use concepts and relationships to describe the domain knowledge; they are the basic unit in ontology construction. The concepts in knowledge tuples will map the concepts in ontology; the loss and error of information in extraction process will be preserved permanently; thus, the tuple extraction model is an important part in ontology automatic construction. According to the structural degree of information sources in IOT environment, we divided the information sources into two categories: unstructured information and structured information. For unstructured information sources, we introduced the context information into tuples and propose the context-information-based (CIB) knowledge-tuple extraction algorithm; for structured information sources, we use the resource description as representative and propose the resource-description-based (RDB) knowledge-tuple extraction model.

3.2.1. Context-Information-Based Knowledge-Tuple Extraction. The unstructured information source refers to information that either does not have a predefined data model or is not organized in a predefined manner, such as text and webpage, which is the most common type of information we contact. The unstructured information provides a knowledge source for tuple extraction. Existing ontology automatic construction models commonly use the natural language processing algorithms to extract knowledge tuples from each sentence of information source. At the same time, they neglect the impact of the context information on tuples. Because the concepts in tuples will map the concepts in ontology, thus, the loss and error of information in extraction process will be preserved permanently. Therefore, we propose a context-information-based knowledge-tuple extraction algorithm. Our algorithm consists of two parts: obtaining the triple knowledge tuples through natural language processing algorithms and then in order to retain the context information in tuple, we extract the core concepts of paragraphs and papers based on the chapter's titles and largest contiguous frequency statistics and then we make the core concepts as the context labels and add them into triple tuples and retain the context information with concept prefix.

In the first part, we use the natural language processing algorithm from Stanford CoreNLP to carry out syntactic analysis for each sentence. We will take raw English language text input and give the base forms of words and their parts of speech, whether they are names of companies, people, and so forth or normalize dates, times, and numeric quantities and we will mark up the structure of sentences in terms of phrases and word dependencies and indicate which noun phrases refer to the same entities. Stanford CoreNLP is an integrated framework, which makes it very easy to apply a bunch of language analysis tools to a piece of text. Its analyses provide the foundational building blocks for higher-level and domain-specific text understanding applications. For example, for the sentence: “Common data formats include XML and json.” The Stanford parser produces the following representations in Figure 2.

Based on the syntactic analysis consequence above, we use Trunk (Trunk provide rules to extract subject-predicate-object tuple from syntactic analysis consequence) to extract triple tuples. In this example, we extract four tuples from syntactic analysis on the right of Figure 2. These tuples use the subject-predicate-object structure to carry knowledge and provide basic units for ontology construction. But each tuple in this part only carries the knowledge from one sentence,
International Journal of Distributed Sensor Networks

Based on the research on IOT scientific papers and technical documents, we can use the tuples in Figure 2 as examples; xml is a data format, but we will never know if this format is used in sensor or gateway. Actually, the concept of a word is determined by the entire paper; thus, we introduce the context lab to retain paper’s information into tuples.

In the second part, we extract multilevel core concepts in a paragraph and make them as prefix for concepts in tuples to retain context information. Here we can also use the long tuple to retain the context information in context. We use two tuples device-has-name-is-sensor and location-has-name-is-china, for example, “device” and “location” are the core concepts in two paragraphs; it has the same redundancy compared with (device_name)-is-(device_sensor) and (location_name)-is-(location_name_china), but consider the common ontology format is owl and it inherited the triple description framework from rdf, thus, it will split the long tuple “device-has-name-is-sensor” and “location-has-name-is-china” into four triple tuples: “device-has-name,” “name-is-sensor,” “location-has-name,” and “name-is-china.” Because the center concept “name,” we will get the wrong inheritance between “location” and “sensor.” Thus we give up the long tuple and use the way of concept prefix in tuples. The models to extract core concepts from unstructured documents can be divided into two types, The first type makes the original chapter titles in documents as core concepts. When a document has clear chapter titles, titles are the summary of paragraphs by author. The parent-child relationship between titles are the reflection of expert knowledge which have high reliability. At the same time, the extraction process is simple and has a small computation overhead. The second type is to extract core concepts from chapters through data mining. Because some documents do not have chapter titles, we need to use data mining to extract the core concepts. The centrality of graph to extract the core concepts from concepts map are commonly used. We use the degree centrality to extract 62 core concepts from 5 IOT scientific papers; more than 70% core concepts are the same with the chapter titles; thus, degree centrality can be used as a supplementary means to extract the core concepts. We use these core concepts as context labs and add these labs into concepts in tuples as multilevel prefix. Through the research on IOT scientific papers and technical documents, more than 95% of the levels of document sections are less than 5. In order to obtain the complete context information of each concept, we will extract core concepts for each paragraph based on the multilevel chapters and construct multilevel prefix for each concept in tuples. At the same time, the calculated amount to maintain the maximum level is bearable (it will not cause the change in computational complexity).

The titles of upper and lower sections meet the one-to-many relationship (parent-child relationship), in order to provide more information for the following similarity calculation, the concepts’ prefix can use ordered vector to retain the parent-child relationship information. Use the sentence in Figure 2 as an example; we can calculate the concept with largest contiguous frequency in second-level chapter as “sensor” and first-level chapter as “IOT.” We can extract the concepts in triple tuples as in Figure 3.

We can achieve the CIB knowledge-tuple extraction as in Algorithm 1. At the same time, we open a test interface in our prototype system to extract tuples from unstructured information sources: http://42.121.104.18:8002/elements/show/nlpTuple.

3.2.2. Resource-Description-Based Knowledge-Tuple Extraction. IOT environment includes various resource descriptions, which are a type of structured documents. Resource description has been used to describe various IOT resources such as device and data; we list a resource description for sensor in Figure 4; it describes various attributes and values for a Temp sensor. We use the Xively (previously Pachube, one of the most famous IOT platform) as an example; it has more than twenty thousand of resource descriptions and exceeds the number of characters in other pages in this website. This helps to obtain a more comprehensive knowledge for tuple extraction. Existing ontology automatic construction models commonly neglect this type of information source. Therefore, we propose a resource-description-based knowledge-tuple extraction model. Compared with unstructured information, resource description has a tree schema to describe the structure of documents and uses the key-value pair like “<Name>Temp sensor</Name>” in Figure 4. The tree schema has been constructed by the IOT experts and provides an explicit superior-subordinate relationship; at the same time, we can extract subject-predicate-object tuples conveniently based on the key-value pair. According to statistical analysis, the vocabularies in structured information are more concise and close to the topic. This will reduce the redundancy for ontology from uncorrelated words.

Figure 4 is a resource description file in xml format; based on the W3C’s definition document, xml is a markup language with key-value model and tree structure. An xml element is everything from (including) the element’s start tag to (including) the element’s end tag; xml elements can have attributes in start tag. The attributes provide additional information about an element; the key-value model of xml defines the relationship between element and value; the tree structure of xml defines the parent-child and brothers relationships between elements; it determines the one-to-many relationship (parent-child relationship in data structure) between upper and lower elements in resource description. Based on the research of resource description files in existing
IOT platforms, the upper and lower elements in resource description mostly have the parent-child or concept-property relations. Considering the property is a partial description of the concept, we can build parent-child relationship between upper and lower elements.

We can divide the elements of resource description into four categories: tag-name (like Sensor and Name), this kind of elements is determined by xml schema and has a tree structure; in the same xml schema, tag-name and its structure are relatively stable; tag-value (like Temp sensor and Huawei), this kind of element is determined by specific resource and often different between different xmls; tag-property (like Tags and ID), this kind of element is determined by xml schema and its structure is relatively stable in the same schema; tag-property-value (like Temp and 12), the same as tag value, the elements are determined by specific resource and are often different between different xmls.

Based on the elements category above, we can analyze the relations between elements and divide the relations into four kinds: hierarchical relations between tag-names, they have been determined by the structure of xml schema; one-to-one correspondence between tag-name and tag-value, they have been determined by specific resource; one-to-many correspondence between tag-name and tag-property, they have been determined by the structure of xml schema; one-to-one correspondence between tag-property and tag-property-value, they have been determined by specific resource. By integrating tag-name ⇔ tag-property and tag-property ⇔ tag-property-value, we can get three kinds of relations and define three rules to extract tuples from xml.

(1) The relationship between upper and lower tag-names is Subclass. For example, for the tag-name "Sensor" and tag-name "Name" in Figure 4, we can extract ternary tuple "Sensor has_subclass Name."

(2) The relationship between tag-name, tag-property and tag-property-value in the same level is corresponding property. For example, for the nodes "Sensor," "Tags," and "Temp" in Figure 4, we can extract ternary tuple "Sensor has_Tags Temp."

(3) The relationship between tag-name and tag-value in the same level is Value. For example, for the tag-name "Name" and tag-value "TempSensor" in Figure 4, we can extract ternary tuple "Name has_value Temp_sensor."

Based on the extraction rules above, we can extract triple tuples from resource description files. But triple tuples only retain the relationship between two layers and give up the multilevel parent-child relationships in resource description; this results in a loss of context information. Thus we need to add the context information into triple tuples and retain the context information with concept prefix. We extract the fourteen tuples in Figure 4 as in Figure 5.

We can achieve the RDB knowledge-tuple extraction algorithm as in Algorithm 2. At the same time, we open a test interface in our prototype system to extract tuples from XML: http://42.121.104.18:8002/elements/show/xmlTuple.

### 3.3 Tuple Normalization.

Tuple normalization converts words from tuples into normal format that can be used by the subsequent steps in ontology construction. It reduces inflected words to their stems, achieves synonyms of each word for each tuple, and finally normalizes the expression of synonyms that are the basis for further operation. In this study, the tuple normalization involves the two processes of word stem and synonym.

(1) Word stem: stemming is the process of reducing inflected (or sometimes derived) words to their stem, base, or root form. For example, measure is the stem for {measuring, measured, measurement}. Stemming of the remaining words is to remove affixes (i.e., prefixes and suffixes), such that the extraction of
3.4. IOT Upper Ontology. Upper ontology (also known as a top-level ontology or foundation ontology) is an ontology which describes very general concepts that are the same across one domain. Selecting an appropriate existing domain ontology as the upper ontology will make full use of knowledge in existing domain ontology. At the same time, the different ontologies which face different systems will have a unified upper ontology. The concepts in different ontologies can be built relations based on the unified upper ontology; it realizes the knowledge sharing across different systems.

In this section we collect and collate existing IOT ontologies and formulate an assessment program and select the appropriate domain ontology as our IOT upper ontology in construction. IOT ontology has a rapid development in recent years. W3C and other organizations and individuals construct a dozen of upper ontology; all these ontologies are not for a specific IOT system; they describe IOT concepts and relationships from different angles. These ontologies generally make the "device" and "observation" as their core concepts, then expand the concepts based on the device type, capacity, properties and observation type, precision, and so on. We collected the famous IOT ontology in Table 1 and evaluate them in five aspects (key concepts, author, status, complexity, and cited). The "key concepts" can reflect the emphasis and scope of concepts for an ontology; we will select the upper ontology based on coincident degree between "key concepts" and our IOT system. The "author" reflects the authority of ontology. The "status" of ontology includes developing, maintaining, and ceasing; it will reflect...
**Algorithm 2: RDB tuple extraction.**

```ruby
Input: xml
Output: context_tuples
xml.each do |element|
  (from element_i to element_M)
  tuple = xml_rules(element)
  Extract context_lab into tuple.
end
```

**Table 1: IOT ontology.**

| Ontology name | Key concepts                                      | Author          | Status     | Complexity | Cited |
|--------------|--------------------------------------------------|-----------------|------------|------------|-------|
| SSN          | Stimulus, sensor, and observation                 | W3C Incubator   | Maintained | Complicated | Yes   |
| CSIRO        | Sensor and process                                | Michael Compton | Developing | Simple     | None  |
| MMI          | Device, capability, and property                  | Luis Bermudez   | Developing | Ordinary   | None  |
| CESN         | Sensor and physical property                      | Holger Neuhaus  | Cease      | Simple     | None  |
| SWAMO        | Platform, process, and observation                | John Graybeal   | Developing | Complicated | Yes   |
| A3ME         | Device, data, service, and capability             | Arthur Herzog   | Maintained | Simple     | None  |
| OnToSensor   | Sensor, capability, and measurement               | Danh Le Phuoc  | Cease      | Complicated | Yes   |
| OBOE         | Observation, context, value, and measurement      | Kevin Page      | Maintained | Simple     | None  |
| SeReS        | Feature and result and observation                | Krzysztof Janowicz | Developing | Complicated | Yes   |
| SemSOS       | Observation, process, feature, and phenomenon     | Cory Henson     | Maintained | Ordinary   | None  |
| Sensei O and M | Observation, data, process, and service         | Payam Barnaghi | Cease      | Simple     | None  |
| OOSTethys    | Process, system, and observation                  | Luis Bermudez   | Developing | Ordinary   | Yes   |
| SERONTO      | Abstract, physical, and reference                 | Laurent Lefort | Cease      | Complicated | Yes   |

We analyze Table 1 and get conclusions as follows. These domain ontologies mostly make “sensor” and “observation” as their core concepts. In terms of author, only the SSN has been built by authoritative W3C Incubator; other ontologies all have been built by individual; there are four ontologies maintained state; they have a high degree of perfection and stable maintenance. In the aspect of complex, we tend to the ontologies with complicated ratings, these ontologies have a fine-grained description for domain knowledge and have been built with an integrated hierarchy. In the aspect of quote, there are six ontologies that have been quoted by other ontologies and systems; they have a high degree of recognition. Taking these conditions above, we choose the SSN [20] ontology which has been built by W3C Semantic Sensor Network Incubator group as our upper ontology. Central to SSN is the Stimulus-Sensor-Observation ontology design pattern. The pattern links sensors, what they sense, and the resulting observations, enveloping three of the four perspectives—the missing system perspective is more about system organization and deployments than sensing but clearly links the pattern. The SSO has been developed as minimal, common ground for heavy-weight ontologies for the Semantic Sensor Web, as well as to explicitly address the need for light-weight semantics in the Linked Data cloud.

Upper ontology will serve as the initial ontology for ontology automatic construction; we will supplement the upper ontology with normalized knowledge tuples through algorithms in the next section. Increasing the scope of ontology knowledge gradually achieves the semantic representation for IOT system finally.

### 3.5. Similarity Algorithm

In this section, we need to calculate the similarity between concepts in tuples and merge the similar concepts to constitute a connected IOT ontology. Because the knowledge tuples in tuple extraction module are discrete; thus, the tuples cannot constitute a knowledge network and cannot be suited for human or machines to understand the relevant knowledge of a concept. Existing approaches use the linguistic similarity of vocabularies to merge synonyms and neglect the scene that the same word in different contexts has different meanings and represents different concepts. Taking into account that we introduce multilayer context labs into concepts in Section 3.2, thus, we propose an aggregation similarity algorithm (TSB) to calculate the similarity between concepts.

Similarity (ontology alignment) is an important research point in ontology construction; it will affect the accuracy of the concepts and relations in ontology directly. In general
the similarity measure can be categorized in lexical, linguistic, and structural measures.

1. In lexical measures, there are a number of string distance methods such as Levenshtein distance [22], Jaro-Winkler distance [23, 24], and smoa distance [25]. And based on [25], the smoa distance is shown to be the most performing distance for the ontology alignment problem (the smoa distance takes into account both commonalities and differences characterizing the entities at issue). We took into account the following two questions: the relations between words string similarity and words similarity, computational complexity. First the words string similarity cannot fully reflect the similarity between words, such as apply, application, and apple. With smoa distance, apple has been closer to apply. But in fact, apply and application have the same stem “apply” and they are more similar. In addition, smoa distance increases the amount of calculation in a large number. Thus based on the research of two aspects above, we used the stem treatment for all words and then with the fully matched comparison, we can get the similarity relations between lexical effective.

2. Linguistic measure computes the similarity between ontology entities by considering linguistic relations such as synonymy and hyponymy. We used the synonym treatment which based on WordNet dictionary deals with the linguistic measure in Section 3.3.

3. The structural similarity measure generally refers to the hierarchy distance measure [21] (based on the subsumption relation between classes in ontologies). And it solved the scene that the same word in different contexts has different meanings and represents different concepts; we proposed two knowledge-tuple extraction methods in Section 3.2 and these two extraction methods all retained the structural information in knowledge-tuple. Based on these above, we propose a tuple-structure-based similarity algorithm (TSB) in Section 3.5.2.

Our ontology construction model is an incremental construction model. We iterate the ontology through adding new knowledge tuple to existing ontology constantly. Because the SSN (upper-ontology) covers the general concepts that are the same across IOT domain, our automatic construction model makes SSN as the initialize ontology, and then expands the concepts and relations through extracting new knowledge tuples. When we expand the existing ontology, we need to identify whether the concept from new tuple can be merged with existing concepts in existing ontology or added as a new concept based on similarity calculation between concepts. In structure similarity calculation, we transform the concepts with multilevel context labs into vectors and assess the structure similarity between vectors. Before we compare the structural similarity, we need to extract contrasted concepts based on linguistics from existing ontology. Therefore, we can divide the structure similarity algorithm into the following two parts: making the linguistic similarity as metrics and extract a contrasted concept-set in existing ontology which is similar to the concepts in new tuple, and then building a contrasted vector group based on the concept-set and making the concept in new tuple as object vector and using the longest common subsequence (LCS) between object vector and contrasted vector to get structural similarity part between two vectors. Through calculating information content (IC) in structural similarity part, we can quantify the structural similarity between two vectors and provide evidence for merging similar concepts. Through merging the similar concepts and adding new concepts, we can iterate the IOT ontology constantly.

3.5.1. Contrasted Concepts Extraction. Contrasted concepts are the concepts of existing ontology which are linguistically similar to the object concept of new tuple; they will be used to calculate the structural similarity. Take into account that we add multilayer context labs into concepts in Section 3.2 and normalize the vocabularies in Section 3.3, thus, when we calculate the linguistics similarity between object concept and concepts in existing ontology, we only need to estimate whether the concepts have the same postfix. If two concepts have the same postfix, we consider that they are in linguistics similarity. Through extracting the contrasted concepts in existing ontology, we can calculate the structural similarity between object concept and contrasted concept, then merge the similar concepts or supplement new concepts into ontology.

Based on the extraction algorithm in Section 3.2, each new tuple is composed of two types of elements: concept with hierarchy labels and relationship. We use the tuple (A_B_C1)-(P1)-(A_B_C2), for example; it contains two concepts “A_B_C1” and “A_B_C2”, each concept has two-level context lab “A” and “B”. We use the “A_B_C1” as object concept. Because the linguistic similarity only relates to postfix “C1”; thus, we will extract the concepts in existing ontology which have the same postfix “C1” and construct the contrasted concept-set. (In order to support the structural similarity comparison, when we add a new concept into ontology, we need to retain the context information for this concept in ontology. Thus we will assign a unique identifier for each concept in ontology. One unique identifier could correspond several express forms and each form will have the same postfix.) In order to facilitate the structure similarity calculation, we transform the object concept and contrasted concept into ordered vectors. Because one object concept could correspond to several contrasted concepts in ontology and each contrasted concept could have several express forms, thus the contrasted concept-set can be represented by a two-dimensional vector group. We use the object concept “IOT_sensor_name” as an example, we assume that there are three concepts named “name” with different “ids” in existing ontology; each concept has several express forms and we list the vector group in Figure 6. Each row in the vector group reflects various express forms for a concept in ontology.

Through the combination and calculation above, we transform the structural similarity comparison between concepts in new tuple and existing ontology into the comparison
between object vector and arbitrary vector in a contrasted vector group.

3.5.2. Vector Similarity Algorithm. Through constructing the contrasted vector in Section 3.5.1, we transform the structure similarity comparison between new tuple and existing ontology into similarity comparison between object vector and arbitrary vector in a contrasted vector group. In order to integrate structural similarity into similarity algorithm, we use the longest common subsequence (LCS) between vectors to get structural similarity part between two vectors. Through calculating information content (IC) in structural similarity part, we can quantify the structural similarity between two vectors and provide evidence for merging similar concepts in the following.

The longest common subsequence (LCS) of two vectors states the similar parts of information provided by the two vectors. A proper quantification of the LCS of vectors improves the degree of structural similarity between vectors. References [26–28] put forward their own similarity algorithm that combine LCS and information content (IC). Reference [29] proposed evaluating the IC of the least common subsumer of the compared concepts (LCS($c_1$, $c_2$)) as the representative of this shared information. The LCS of a pair of concepts is the most specific common ancestor that subsumes them; it is found in the taxonomy to which they belong. If the two concepts are not taxonomically connected and the LCS does not exist, they are considered maximally different. Otherwise their semantic similarity is computed as the amount of IC provided by the LCS.

Consider

$$\text{sim}_{[29]}(c_1, c_2) = \text{IC} \left( \text{LCS} \left( c_1, c_2 \right) \right).$$

(1)

With [26]'s metric, any pair of concepts with the same LCS will result in exactly the same similarity value. To better differentiate concepts, both [27, 28] also consider the IC of the compared terms into the equations. Reference [27] measures the similarity as the ratio between the common information between concepts (i.e., IC(LCS)) and the information needed to fully describe them (i.e., the IC of each concept alone). Reference [28] proposed calculating the concept distance (the opposite of similarity) as the difference between the IC of each concept and the IC of their LCS.

Consider

$$\text{sim}_{[30]}(c_1, c_2) = \frac{2 \times \text{IC}(\text{LCS}(c_1, c_2))}{(\text{IC}(c_1) + \text{IC}(c_2))},$$

$$\text{sim}_{[26]}(c_1, c_2) = \left( \text{IC}(c_1) + \text{IC}(c_2) \right) - 2 \times \text{IC}(\text{LCS}(c_1, c_2)).$$

(2)

Here we need to calculate the similarity between tuple vector and arbitrary vector in the vector group. Through LCS($v_1$, $v_2$), we can get the number of the same concepts directly and it reflects the same portion of two vectors. But any pair of vectors with the same LCS will result in exactly the same similarity value. Therefore we need to measure the ratio between the common information and all information. In (1) all information is a fixed value and reflects the information of universal set. But in our algorithm all information only reflects the total information of two vectors and it will be changed with vectors. We can get vector similarity algorithm as follows:

$$\text{IC}(\text{LCS}(v_1, v_2)) = -\log \left( \frac{\text{LCS}(v_1, v_2)}{v_1 \cup v_2} \right).$$

(3)

Then we need to merge similarity concepts based on vector similarity and threshold. According to the contrasted vector group in Figure 6, each row in the vector group reflects various express forms for a concept in ontology. We can divide the similarity computation results into three types: first type, there is not a vector similar to object vector in contrasted vector group, we will add the object concept into ontology as new concept; second type, there is only one row which has vector similar to object vector in contrasted vector group, we will merge object concept with the concept which corresponding to the one row, at the same time we add the new express form into this row in group; third type, there are more than one row which have vector similar to object vector in contrasted vector group, we need to merge all similar concepts and merge the corresponding express forms.

In the third type, the new concept will change the partial structure of existing ontology. The similarity threshold defines the maximum similar distance between vectors, but each concept in ontology does not define its absolute standard express forms (centre); therefore, the similar space for a concept is a dynamic space with the change of concept's express forms. With the expanding of express forms for a concept, the similar space is also increasing. We can get the edge distance between multiple concepts based on the similar space of each concept in ontology; when the edge distance is smaller than the maximum diameter of the similar space of new express forms, the new express forms may cause the combination of multiple concepts in ontology and this is the occurrence of third type. With the extension of concepts and express forms in ontology, the edge distance between polysemous concepts will be reduced gradually. Here some noise tuples (Because of wide range of information source, it is difficult to ensure that each tuple complies with the basic knowledge of IOT domain; some artificial
nonstandard information will be used as tuples to iterate IOT ontology) may cause the collapse (since the individual noise information led to the concept of the ontology polysemy merge, resulting in loss of information) of partial ontology concepts. This paper only presents this possible problem and does not make a detailed exposition. In future studies we intend to supplement the merger control factor to control the merging between polysemy concepts in ontology.

We calculate the complexity of the structure similarity algorithm (Algorithm 3). When adding a concept, we suppose that the number of concepts in ontology is \( m \), the number of concepts’ expression is \( n \), and the length of the expression is \( t \). We can calculate the time complexity as \( O(m \times n \times \log(t)) \). Consider the length of the expression does not increase with the expansion of ontology; thus, \( \log(t) \) can be considered as a constant, and the time complexity can be simplified as \( O(m \times n) \).

4. Prototype and Evaluation

In this section, we build a prototype system and collect files from six IOT platforms. These files include platform introductions, technical documents, and resource descriptions. We put these files and SNS ontology into prototype system, select an appropriate threshold, and get the IOT ontology. Then we evaluate the ontology based on the experts’ knowledge and evaluating system.

4.1. Prototype System. We use ruby on rails to build the prototype system and deploy it in ubuntu environment with two 2.26 GHz Xeon processor and 4 G of memory. Prototype system includes two main modules: basic components and complete process. We have developed more than ten atomic functions in basic component. Include searching synonym, hypernym, and hyponym based on WordNet; removing stopwords; achieving stem based on [19]; segmenting Chinese Word; calculating tags based on random walk; extracting knowledge tuple based on stanford-core-nlp, XML, and ontology; building ontology in owl format based on knowledge tuples. All these functions have been open with restful API and we list them in Table 2 (detail in http://42.121.104.18:8002/api/api.pdf).

By invoking the APIs in Table 2, we assembled several types of automatic construction processes which adapt to different needs in complete process module. These processes include the upper-ontology-based approach for automatic construction of IOT ontology in this paper (http://42.121.104.18:8002/services/show/two). Users can enter the url of upper ontology, xml, and text in web page; then system will build an ontology by invoking all atomic functions in Section 3.

4.2. Data Sources and Construction Result. We collect data source files which include platform introductions, technical documents, and resource descriptions from six IOT platforms. These files are expressed by text, web page, xml, or json (as a structured document, json can be converted to xml) and we classify the files based on data format and application scenarios in Table 3. The websites of six platforms are listed below the table: Xively, previously Pachube, is a web service which enables users to connect devices into platform and share data; Cloudsensing is our WOT platform; we put in various sensors, controllers, and open data in each layer; Evrythng, Thingspeak, and Yeelink are all IOT applications which store and retrieve data from things using RESTful API. Exosite has a function to generate strategy; we can formulate
Table 2: Basic components API.

| Name                  | URL                                                                 | Method | Input  | Output   |
|-----------------------|----------------------------------------------------------------------|--------|--------|----------|
| Word segmentation     | http://42.121.104.8002/elements/show/rmmseg                         | POST   | text   | words    |
| Stem                  | http://42.121.104.8002/elements/show/stem                          | GET    | word   | stem     |
| Synonym               | http://42.121.104.8002/elements/show/synonym                        | GET    | word   | synonyms |
| Hypernym              | http://42.121.104.8002/elements/show/hypernym                       | GET    | word   | hypernyms|
| Hyponym               | http://42.121.104.8002/elements/show/hyponym                        | GET    | text   | hyponyms |
| Remove stopwords      | http://42.121.104.8002/elements/show/stopword                      | POST   | text   | words    |
| XML tuple             | http://42.121.104.8002/elements/show/xmlTuple                      | POST   | urls   | tuples   |
| NLP tuple             | http://42.121.104.8002/elements/show/nlpTuple                      | POST   | text   | tuples   |
| OWL tuple             | http://42.121.104.8002/elements/show/owlTuple                      | POST   | urls   | tuples   |
| OWL packaging         | http://42.121.104.8002/elements/show/owl                            | POST   | tuples | url      |

Table 3: Data sources.

| Resource               | Type                   | C1 | C2 | E1 | E2 | T | Y | Total |
|------------------------|------------------------|----|----|----|----|---|---|-------|
| Text                   | Platform introduction  | 2  | 1  | 1  | 2  | 1 | 7 | 7     |
|                        | Technical document     | 3  | 1  | 1  | 1  | 1 | 7 | 7     |
|                        | Smart home             | 39 | 28 | 12 | 6  | 28| 23| 136   |
|                        | Smart agriculture      | 19 | 2  | 9  | 30 |    |   |       |
|                        | Energy monitor         | 57 | 8  | 13 | 7  | 13| 78|       |
| Resource description   | Air quality            | 64 | 23 | 17 | 7  | 12| 25| 148   |
|                        | Ocean Observations     | 13 |    |    |    | 13|   |       |
|                        | Weather station        | 57 | 6  | 13 | 14 | 7 | 97|       |

C1: Xively: https://xively.com/, C2: Cloudsensing: http://wot.cloudsensing.cn/  
E1: Evrythng: https://dev.evrythng.com/, E2: Exosite: http://exosite.com/  
T: Thingspeak: https://www.thingspeak.com/, Y: Yeelink: http://www.yeelink.net/.

Table 4: Number of information sources and ontology.

| Stage | Number of information sources | Number of elements in ontology |
|-------|-------------------------------|--------------------------------|
|       | Text                          | Concept                        |
| 1     | 3                             | 100                            | 1376                           |
| 2     | 6                             | 200                            | 2197                           |
| 3     | 9                             | 300                            | 2507                           |
| 4     | 12                            | 400                            | 2847                           |
| 5     | 14                            | 502                            | 3045                           |

The six IOT platforms have a similar IOT platform architecture; open resources in three levels: gateway, device, and data. We collected 14 texts and 502 resource description files from the platforms. In order to get the relations between the number of imported information sources with the IOT ontology, we divide all information sources into five parts and import them into prototype gradually. At the same time, we make the SSN as upper ontology and it will serve as initial ontology for ontology automatic construction, the concepts in SSN describes very general concepts that are the same across one domain; thus, we consider the concepts in SSN as unique (reflect the same concept in different contexts) in IOT domain, and the linguistic similarity will reflect the concept similarity. We can get the relation-table as in Table 4.

Based on Table 4, we can get the conclusions as follows. The number of concepts in ontology is increasing with the number of imported information sources, but the growth rate is reducing gradually. Because of the unceasing expansion of ontology, there are a growing number of concepts in ontology. At the same time, the probability of finding a structure-similar concept in existing ontology with the new concept in tuple is increasing constantly; therefore, the growth rate of concepts is reducing. After importing the 14 texts and 502 resource description files, the IOT ontology contains 3045 concepts totally. These concepts cover general concepts in IOT domain such as “gateway,” “device,” and “observation.” At the same time, they also reflect the features of six IOT platforms, such as “event” and “alert.” We use “device” and “observation” as centre to unfold related parts of ontology with protégé and get the views in Figures 7 and 8.

4.3. Ontology Evaluation. In order to evaluate the performance of the construction approach, we investigate the general method of ontology evaluation and define an assessment scheme to estimate the effectiveness of IOT ontology. There are three main approaches to ontology evaluation.

(1) Gold standard evaluation: this approach compares an ontology with another ontology that is deemed to
be the benchmark. Typically, this kind of evaluation is applied to an ontology that is generated (semi-automatically or according to a learning algorithm) to assess the effectiveness of the generating process. Maedche and Staab [30] give an example of a gold standard ontology evaluation and they propose ways to empirically measure similarities between ontologies both lexically and conceptually based on the overlap in relations. These measures determine the accuracy of discovered relations generated from their proposed ontology learning system compared with an existing ontology but are not so useful outside the domain of ontology learning because if a known gold standard ontology exists then there is no need to evaluate other ontologies.

(2) Criteria-based evaluation: this approach takes the ontology and evaluates it based on criteria [18] such as consistency, completeness, conciseness, expandability, and sensitivity. It depends on external semantics to perform the kind of evaluation that only humans are currently able to do, since it is difficult to construct automated tests to compare ontologies using such criteria [11]. These criteria focus on the characteristics
of the ontology in isolation from the application. So, while ontology criteria maybe met, it may not satisfy all the needs of the application even if some needs may correspond with the ontology criteria.

(3) Task-based evaluation: this approach evaluates an ontology based on the competency of the ontology in completing tasks. In taking such an approach, we can judge whether an ontology is suitable for the application or task in a quantitative manner by measuring its performance within the context of the application. The disadvantage of this approach is that an evaluation for one application or task may not be comparable with another task. Hence, evaluations need to be taken for each task being considered.

Based on the above three types of evaluation methods, we can get conclusions as follows: existing ontologies describe the basic concepts of IOT in a coarse-grained way, but because of the limitations of the artificial construct speed and rapid development of IOT technology, these domain ontologies lack a description for system features and emerging concepts, and they are not suitable for direct use in our specific IOT system. Therefore they cannot support Gold standard evaluation as the Gold standard; the IOT ontology which we build contains thousands of concepts and relationships; it is too onerous for manual evaluation. And our IOT system resources are increasing rapidly with the equipments access to our IOT system; the ontology also is expanding continually. We need an automated ontology evaluation method to evaluate the existing ontology in real time; thus, criteria-based evaluation is not suitable for our assessment program; ontology can be used in many IOT scenes such as heterogeneous resource search and service development, and ontology-based semantic search is the basic task of various scenes; therefore, we can use the semantic search as the task for task-based evaluation, making precision and recall as the evaluation index, and achieving the automatic ontology evaluation. We built the task-based evaluation architecture in Figure 9. Evaluation architecture mainly contains the following three steps: importing a keyword from xml file which is to be searched, carrying out semantic expansion in IOT ontology based on the keyword, and calculating the precision and recall between semantic expansion concepts and all vocabularies in xml file which is to be searched.

(1) In the first step, we choose the resource description files as the object to be searched. Resource description is a kind of structured xml document, widespread in the IOT systems, and is used to describe various types of open resource. The resources search in IOT systems are performed for resource description files, thus making resource description as the searched files that will conform to the IOT scene. We select 1~3 words from resource description to simulate the user’s search input and put them into second step. At the same time, all vocabulary set in resource description will be imported into third step as an ontology evaluation input.

(2) In the second step, we achieve several concepts in the ontology which are corresponding to the keywords in first step, and then get their parent-child concepts as semantic expand set. Then we make the semantic expand set as an input and import it into third step.

(3) In the third step, we evaluate the ontology based on all vocabulary set in resource description (G_w) and semantic expand set (G_c) and get the precision and recall index. Consider

\[
\text{Precision} = \frac{G_w \cap G_c}{(G_w \cap G_c) \cup (G_c - G_w)},
\]

\[
\text{Recall} = \frac{G_w \cap G_c}{(G_w \cap G_c) \cup (G_w - G_c)}.
\]

In order to compare the ontology-based semantic search effects with different ontologies, we construct a contrastive ontology with traditional methods. We used the information sources in Table 3 to ensure knowledge consistency. Then we use the natural language processing algorithms to extracted triple tuples from information sources. Through merge synonyms in tuples with the linguistic similarity of vocabularies, we get the connected contrastive ontology.

We randomly select 100 resource description files from information sources and make them as searching objects. According to statistical analysis with google, the situation which users enter 1~4 keywords to search an object cover more than 95%. Therefore, we will randomly select 1~3 keywords in each resource description and import the keywords into our ontology and contrastive ontology. Through getting the parent-child concepts within 3 jump and direct properties with keywords in ontologies, we can get two semantic expand sets as G_w1 (our ontology) and G_w2 (contrastive ontology). Then we collect all vocabularies in one resource description as G_w, and calculate the precision and recall based on formula (4). We divide the evaluation table based on the number of

| Keywords | Precision | Recall |
|----------|-----------|--------|
| 1        | 87.43%    | 41.37% |
| 2        | 87.36%    | 55.29% |
| 3        | 87.18%    | 70.54% |
| 4        | 87.74%    | 78.83% |

Table 5: Precision and recall.
keywords and two ontologies, calculate the average values of 100 resource descriptions, and get precision-recall in Table 5.

Based on Table 5, we can get the conclusions as follows. Using the same ontology, precision remains unchanged as the keywords increase. The precision of our ontology is 11% higher than the contrastive ontology’s. With the same ontology, recall is increasing with the number of keywords, but the growth rate is reducing gradually. With the same number of keywords, the recalls of two ontologies are the same. In the aspect of precision, because of the contrastive group using triple tuples and linguistic similarity to construct the ontology, some concepts inherited irrelevant upper concepts. We use a couple of tuples as example: device-has-name-is-sensor and location-has-name-is-china, if we use the way of triple tuples, we can get device-has-name, name-is-sensor, location-has-name, and name-is-china. After merging the concept “name” based on linguistics, the “sensor” will inherit “device” and “location”; this will cause the decline in precision; in the aspect of recall, with the increasing of keywords’ number, we can get more parent-child concepts within 3 jump and direct properties in ontology; thus, the recall is increasing. At the same time, with the increasing of keywords’ number, it is more possible to have an intersection between concept sets with different keywords. We reveal this situation in Figure 10; red point represents the keyword and blue point represents the relevant concept in ontology; because of the intersection between sets, the number of blue points does not have a linear growth with red point. Therefore the growth rate is reducing gradually.

5. Conclusions and Future Work

An IOT ontology can help users and computers to grasp the knowledge of IOT system. Rapid, accurate construction of ontology has become an important topic for researchers working on semantic web. This paper proposed a new architecture for ontology automatic construction: selected the appropriate existing domain ontology as the upper ontology, made full use of knowledge in existing domain ontology and reduced the number of extracting new concepts and relationships; proposed the context-information-based knowledge-tuple extraction algorithm and triple-structure-based similarity algorithm and ensured the accuracy of concepts and relationships in ontology; proposed the resource-description-based knowledge-tuple extraction model and made full use of the structured information in IOT systems.

We constructed a semantic search task-oriented ontology evaluation architecture and evaluated different IOT ontology construction architectures from two aspects: precision and recall. The construction architecture in this paper can improve the search recall well. Our opinions and methods in this paper could be a starting point for automates construction which utilize various information source in IOT.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The work is supported by the National 3rd Key Program project (no. 2011ZX03005004-02) and Project on the Architecture, Key technology research and Demonstration of Web-based wireless ubiquitous business environment (no. 2012ZX03005008).

References

[1] S. Steffen and S. Rudi, Eds., Handbook on Ontologies, International Handbooks on Information Systems, 2009.
[2] S. Hachem, T. Teixeira, and V. Issarny, “Ontologies for the internet of things,” in Proceedings of the 8th Middleware Doctoral Symposium (MDS ’11), 2011.
[3] M. Compton, P. Barnaghi, L. Bermudez et al., “The SSN ontology of the W3C semantic sensor network incubator group,” Web Semantics: Science, Services and Agents on the World Wide Web, vol. 17, pp. 25–32, 2012.
[4] http://www.w3.org/2005/Incubator/ssn/wiki/Report_Work_on_the_SSN_ontology.
[5] M. Y. Dahab, H. A. Hassan, and A. Rafea, “TextOntoEx: automatic ontology construction from natural English text,” Expert Systems with Applications, vol. 34, no. 2, pp. 1474–1480, 2008.
[6] R.-C. Chen, J.-Y. Liang, and R.-H. Pan, “Using recursive ART network to construction domain ontology based on term frequency and inverse document frequency,” Expert Systems with Applications, vol. 34, no. 1, pp. 488–501, 2008.
[7] X. Hou, S. K. Ong, A. Y. C. Nee, X. T. Zhang, and W. J. Liu, “GRAONTO: a graph-based approach for automatic construction of domain ontology,” Expert Systems with Applications, vol. 38, no. 9, pp. 11958–11975, 2011.
[8] R. Subhashini and J. Akilandeswari, “A survey on ontology construction methodologies,” International Journal of Enterprise Computing and Business Systems, vol. 1, no. 1, 14 pages, 2011.
[9] I. Niles and A. Pease, “Towards a standard upper ontology,” in Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS ’01), 2001.
[10] C.-W. Shih, M.-Y. Chen, H.-C. Chu, and Y.-M. Chen, “Enhancement of domain ontology construction using a crystallizing approach,” Expert Systems with Applications, vol. 38, no. 6, pp. 7544–7557, 2011.
[11] C.-S. Lee, Y.-F. Kao, Y.-H. Kuo, and M.-H. Wang, “Automated ontology construction for unstructured text documents,” Data & Knowledge Engineering, vol. 60, no. 3, pp. 547–566, 2007.
[12] A. Zouaq, D. Gasevic, and M. Hatala, “Towards open ontology learning and filtering,” Information Systems, vol. 36, no. 7, pp. 1064–1081, 2011.
[13] M. Kamel and N. Aussenac-Gilles, “Ontology learning by analysing XML document structure and content,” in Proceedings of the Knowledge Engineering and Ontology Development (KEOD ’09), pp. 159–165, Madère, Portugal, 2009.

[14] Y. Jung, J. Ryu, K.-M. Kim, and S.-H. Myaeng, “Automatic construction of a large-scale situation ontology by mining how-to instructions from the web,” Web Semantics: Science, Services and Agents on the World Wide Web, vol. 8, pp. 110–124, 2010.

[15] D. Sánchez and A. Moreno, “Learning non-taxonomic relationships from web documents for domain ontology construction,” Data & Knowledge Engineering, vol. 64, no. 3, pp. 600–623, 2008.

[16] R.-C. Chen, C.-T. Bau, and C.-J. Yeh, “Merging domain ontologies based on the wordnet system and fuzzy formal concept analysis techniques,” Applied Soft Computing, vol. 11, no. 2, pp. 1908–1923, 2011.

[17] A. Maedche and S. Staab, “Measuring similarity between ontologies,” in Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, vol. 2473 of Lecture Notes in Computer Science, pp. 251–263, 2002.

[18] http://ckip.iis.sinica.edu.tw/CKIP/engversion/index.htm.

[19] M. Porter, “An algorithm for suffix stripping,” Program, vol. 14, no. 3, pp. 130–137, 1980.

[20] W3C, The SSN Ontology of the W3C Semantic Sensor Network Incubator Group, 2011.

[21] G. Acampora, V. Loia, S. Salerno, and A. Vitiello, “A hybrid evolutionary approach for solving the ontology alignment problem,” International Journal of Intelligent Systems, vol. 27, no. 3, pp. 189–216, 2012.

[22] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Soviet Physics—Doklady, vol. 10, no. 8, pp. 707–710, 1966.

[23] M. A. Jaro, “Probabilistic linkage of large public health data files,” Statistics in Medicine, vol. 14, no. 5–7, pp. 491–498, 1995.

[24] W. Winkler, “The state record linkage and current research problems,” Tech. Rep., Statistics of Income Division, Internal Revenue Service Publication, 1999.

[25] G. Stoiilos, G. Stamou, and S. Kollias, “A string metric for ontology alignment,” in Proceedings of the 4th International Semantic Web Conference (ISWC ’05), pp. 623–637, 2005.

[26] P. Resnik, “Using information content to evaluate semantic similarity in a taxonomy,” in Proceedings of the 14th International Joint Conference on Artificial Intelligence, (IJCAI ’95), C. S. Mellish, Ed., vol. 1, pp. 448–453, Morgan Kaufmann Publishers, Quebec, Canada, 1995.

[27] D. Lin, “An information-theoretic definition of similarity,” in Proceedings of the 15th International Conference on Machine Learning (ICML ’98), J. Shavlik, Ed., pp. 296–304, Morgan Kaufmann, Madison, Wis, USA, 1998.

[28] J. J. Jiang and D. W. Conrath, “Semantic similarity based on corpus statistics and lexical taxonomy,” in Proceedings of the 10th International Conference on Research in Computational Linguistics (ROCLING ’97), pp. 19–33, Taipei, Taiwan, 1997.

[29] OGCSWE, OpenGIS Observations & Measurements Implementation Specification, 2007.

[30] W3C, Web Services Description Language (WSDL) 1.1, 2001.
