http://dx.doi.org/10.5392/IJoC.2016.12.2.024

Semi-Automatic Ontology Construction from HTML Documents: A conversion of Text-formed Information into OWL 2

Chan jong Im
Department of Information and Telecommunication
Pai Chai University, Deajeon, 155-40, Republic of Korea

Do wan Kim
Pai Chai University, Deajeon, 155-40, Republic of Korea

ABSTRACT

Ontology is known to be one of the most important technologies in achieving semantic web. It is critical as it represents the knowledge in a machine readable state. World Wide Web Consortium (W3C) has been contributing to the development of ontology for the last several years. However, the recommendation of W3C left out HTML despite the massive amount of information it contains. Also, it is difficult and time consuming to keep up with all the technologies especially in the case of constructing ontology. Thus, we propose a module and methods that reuse HTML documents, extract necessary information from HTML tags and mapping it to OWL 2. We will be combining two kinds of approaches which will be the structural refinement for making an ontology skeleton and linguistic approach for adding detailed information onto the skeleton.

Key words*: Ontology, Semantic Web, HTML, Natural Language Processing.*

1. INTRODUCTION

The concept semantic web is to develop a web into an intelligent space where all the information and knowledge is machine readable and refineable. Among the technologies recommended by the World Wide Web Consortium (W3C), ontology is one of the most important technologies in representing knowledge and sharing it [1].

There are several well established upper ontologies such as [2], [3] which allows the domain information to be linked and shared through the formal ontologies. However, process of constructing domain ontology is difficult and time consuming even working with an expert of the domain. Moreover, most of the researches on automatic ontology construction are based on fundamental technologies such as XML, RDF(s), and OWL while HTML is left aside despite the abundant information it contains [10]. Thus, researches for the better usage of information which HTML documents contain is needed.

Development into HTML5 has made HTML documents to have more semantic meanings. However, former version of HTML, version 4.01, is still widely used. Its predominance makes it more difficult to make web into an intelligent space. Thus, we propose our method that mainly uses the sequences of list tags that are consist of <*ul*>, <*ol*>, <*li*> tags, to build a skeleton ontology containing information about the relationships among the HTML documents in the domain. In addition, more detailed information is extracted from the tags containing contents and they are added to the skeleton through English syntax analysis using natural language processing tools provided by Stanford University.

2. RELATED WORK

There are several approaches in building ontology from HTML tags. Firstly, the structural analysis approach which has information on simple and direct structural mappings from HTML to OWL. [4] showed the mappings for HTML table, checkbox, radio, select tags into OWL. It proposed the mapping rules for HTML tags and had no problem in OWL Lite validation. [5] tried to classify newly established tags in HTML5 and proposed schema level mapping rules based on the semantic elements and instance mapping rules.

These works are helpful in making the draft of ontologies. However, the established ontologies are mapped from HTML tags without considering the concept of representing the knowledge [1]. The fact that HTML was originally designed for better presentation to humans and the fact that it is not containing any semantic information make these structural mapping methods unreasonable.

* *Corresponding author, Email: dwkim@pcu.ac.kr*
Manuscript received Apr. 28, 2016; revised May. 16, 2016; accepted May. 23, 2016

Assigning semantic information to HTML is a significant task for establishing semantic web. For this purpose, a second approach for building an ontology, linguistic approach was brought about which dealt with syntactical analysis as a part of natural language processing (NLP). One of the most recent methods in analyzing English sentences is done by Part-Of-Speech Tagging [7]. Furthermore, with the return sets of POS tagging elements, several attempts to systematically convert it into triples has been conducted as a part of Open Information Extraction (IOE) in works [8] and [9].

Hwangbo et al [10] used the structural mappings and some NLP tools to make the ontology. It pointed out that fundamental technologies of semantic web that has been recommended by W3C were too much concentrated in XML and RDF(s) which made HTML useless for semantic web. The recommendation of Gleaning Resource Descriptions from Dialects of Languages (GRDDL), a technology of HTML conversion, was noted to have deficiency since it needed valid style sheets and had limitations in making RDFS. Thus, the article proposed procedural steps in making an ontology from HTML tags. These steps contain extracting information from general HTML documents, classifying the tags, rules for extracting data of each tag group, transforming it into triples for both text-formed information and mixed-formed information as a part of structural mapping methods, and analyzing the triples with WordNet [17] and To-and-To web application as a part of a linguistic approach. Though these approaches seemed promising and realizable, they only focused on the ways to map the data contained in a single HTML document. In other words, they were not able to construct an ontology with the information of relationships among other pages in the same domain.

Since HTML documents lack semantic information, the process of refinement was needed in order to extract the data from HTML documents. This is done by using the object Document Object Model (DOM) [12]. The article [13] tried to discover the semantic pattern referred to as 'similarity' in the implicit fixed schema of template-driven HTML documents and automatically generated a semantic partition tree. This was done upon the observation of spatial locality of the contents in template-driven HTML documents. Though their work seemed well working in finding semantic structure of the document, it was limited to the template-driven documents and modification was needed for ontology construction.

Another way to assign semantic information from relationship retrieval which specifically is related to extracting key terms and forming them into a concept hierarchy. Article [14] tried to extract hierarchical relationships based on modified Formal Concept Analysis (FCA) theory [11]. Modified form of FCA theory allowed the attributes of a term to have a certain level of unnecessary values. In other words, it allowed to have some attributes of a term that are not perfectly fitted to form a hierarchy relationship with a chosen threshold value. With the keywords given by the domain experts, attributes of the keywords were chosen with the window size that ranged from one to five. These attributes were rearranged into document-weight vector and were clustered with k-means methods to put all the similar attributes together. The rules of modified FCA theory were applied to the keyword with attribute clusters to form a hierarchical concept.

The article shows a good performance in allocating the words and terms into higher concept. However, it is insufficient to build the ontology with only hierarchical relationships. Also, the problem in choosing keyword and cluster size remained unsolved.

3. PROCEDURES FOR ONTOLOGY CONSTRUCTION

In order to build a domain ontology using OWL 2 which is more expressive than OWL 1 [15], we will be using the combination of structure and linguistic approaches in this paper. Using the combination of two approaches serves as complementary one to the other in building an ontology. For the problem of semantic information scarcity in the case of using structural mapping method, for example, can be supplemented by linguistic approach. On the other hand, the problems in linguistic approach such as concept labeling or limitation on document types, can be solved with comprehensively defined structural rules.

Our goal is to make a basic ontology from general HTML documents, specifically from all HTML documents in the domain. The proposed module is composed of 3 phases which are extracting nodes of trees, mapping onto the ontology, and adding details onto the skeleton ontology. All tags that are used in first phase of our module are shown in Table 1. We left out all the unused tags from the tag classification mentioned in article [10]. In the introduction, it was mentioned that we will be using the tags that are used for easy navigation to make the skeleton ontology for the specific domain. In the case of HTML 5, for example, all tags constructed for easy navigation are grouped together under <*nav*> tag and it holds <*ul*>, <*ol*>, <*li*> tags which are classified in the formal HTML tag classification as *List* in table 1. Specifically, <*ul*>, <*ol*> tags does not hold any values but declares that it is either unordered list <*ul*> or ordered list <*ol*>. The tags that actually hold some values are <*li*> tags. These tags hold <*a*> tags, classified as *Link* category in table 1, which contain values of URL links.

Table 1. Tags used for forming skeleton ontology [10]

Category	HTML Tag
Link	*A*
List	*li, ol, ul*
Paragraph	*P*

We tried to extract hierarchical relationships based on the way it is nested. For instance, tag <*ul*> might contain <*li*> tags but also might contain another set of lists declared by second <*ul*> or <*ol*> placed inside the first <*ul*> tag. In this case, the <*li*> tags nested under first appeared <*ul*> tag are defined to be as parent nodes, and the <*li*> tags in second appeared <*ul*> or <*ol*> are defined to be as child-nodes of the corresponding parent node. We use these inclusion levels of tags to form a hierarchical tree, which is later reformed into the ontology syntax with some rules being implied.

In the third phase of our work, we tried to add detailed information onto the skeleton by extracting content elements from <*p*> tags in category *Paragraph* in table 1 and the corresponding URL values located in <*a*> from the place where <*p*> was being extracted.

In the area of Information Extraction (IE), importance of recognizing the name entities had been highlighted at the Sixth Message Understanding Conference (MUC-6) [20]. Thus, we search for the sentences that contain the name entities which are recognized by the tool Named Entity Recognizer (NER) [21] by Stanford University.

These sentences are analyzed based on English syntax and are transformed into triples which is consist of *Subject, Predicate, Object,* by the tool Open Information Extractor [22] from Stanford. These triples are then mapped onto skeleton ontology. In the following subsection 3.1 and 3.2, the steps required for ontology construction are explained in more detail.

3.1 Extracting tree from *list* tags

In order to make a skeleton ontology, we first extract the necessary tags and form a tree structure before the process of mapping onto the ontology. The depth of the tree is distinguished by the tag <*ul*> and the end tag </*ul*>. Leaf-nodes of a tree are assigned by the tag <*li*> for its corresponding tree depth. In the following subsections, there will be an explanation of the values held in each leaf-node of extracted tree in section 3.1.1 and details of the procedures for extracting nested tags with the depth of tree in 3.1.2.

3.1.1 Node values: The values which each node contains are presented in Fig. 1. Each node is linked with either parent nodes or children nodes which is decided by the sequence of nested tags in HTML documents. Each of the node hold values of <*a href*> tags and pure text values which are used for assigning the text labels when it is being transformed to ontology.

```
public class TreeNode {

  private TreeNode parent = null;
  private List<TreeNode> children = null;
  private Object text;
  private String href="";
```

Fig. 1. Values held in each node

For instance, the root node of the tree will contain child-nodes which are the values of <*li*> tags within the first boundary of <*ul*> tag. Each node will contain its URL value as well as its pure text value. URL value will be saved in 'href' variable and pure text values will be saved in the 'text' variable.

3.1.2 Procedures: We use Jsoup [6], Java API, which is known to have a good control over the HTML tags to refine HTML document and to generate a hierarchical tree. Our aim is to extract the relationship from nested <*ul*> tags with the maximum depth of three (assigning root as depth of zero). Though this API is useful in extracting nested tags, the use is limited to the depth of two. For instance, if there are sequences of lists like <*ul*>-<*ul*>-<*li*>-</*ul*>-</*ul*> and <*ul*>-<*ul*>-<*ul*>-<*li*>-</*ul*>-</*ul*>-</*ul*>, Jsoup is capable of extracting <*li*> tag in the first sequence of lists whereas it needs additional algorithms for extracting <*li*> tag in the second sequence of lists. Thus, we use Jsoup as our base with additional method added on top of it.

The following procedural steps will show how the nodes are extracted and formulate a hierarchical tree.

- Using Jsoup, we filter out the unused tags which are out of the boundary of <*ul*> tags and get only <*ul*> tags and all the tags which are located within the boundary of <*ul*>. We do this by firstly assigning main-page of the website as a root of the tree. We then select the first <*ul*> tag and get the first element which returns the pure sequence of <*ul*> and <*li*> tags without any other tags.

- With the pure sequence of *List* tags, we extract the child-nodes of the root. These are the <*li*> tags within the boundary of the first <*ul*>. To extract these nodes, we select all the tags that have the tree depth of one and contain <*li*> tags.

- Having a list of child-nodes of the root node, we then tried to get the child-nodes of the nodes that we got in the previous step. For simplicity, we will refer the nodes we got in the previous step as parent-nodes and the nodes that we are trying to get as child-nodes. The child-nodes are the <*li*> tags within the boundary of the second <*ul*> tags, in other words, <*li*> tags in tree depth of two. Additional step is required before getting child-nodes which is to recognize whether the parent-node contains child-nodes or not. This is recognized by the sequence of *List* tags. If there are any <*ul*> tags appearing after the parent-node, it is known to have a child-nodes and vice versa. For instance, if there is a sequence of List tags like <*ul*>-<*li*>-<*li*>-<*ul*>-<*li*>-</*ul*>-</*ul*>, first parent-node <*li*> does not have another set of lists nested by <*ul*> tag, whereas the second parent-node <*li*> has the following <*ul*> tag. Thus, first parent-node does not contain any child-nodes and the second parent-node contain child-node of <*li*> nested in the second <*ul*> tag.

- The same rule applies when getting the last child-nodes in the tree depth of three. First it is being recognized whether the parent-node contains any child-nodes, and if there is any <*ul*> nested after the parent-node, all the <*li*> within the corresponding <*ul*> are linked as child-nodes of that parent-node.

3.2 Mapping tags on skeleton ontology and adding detailed information

In the previous section, all *List* tags were linked forming a hierarchical tree. In this section, nodes from the established hierarchical tree are mapped onto basic skeleton ontology with few mapping rules applied to it. Also, more information is extracted from the HTML documents and is added on to the established skeleton ontology. In the following subsections, there will be explanations of implementation of the mapping

rules and the tools used for mapping in subsection 3.2.1, and how the detailed information are added to the skeleton ontology will be explained in section 3.2.2, 3.2.3, and 3.2.4.

3.2.1 Mapping tag nodes on skeleton ontology: Given that all the *List* tags are formulating the hierarchical tree, to transform it into ontology, decision has to be made for each node whether it is an entity, class or individual before the actual mapping is conducted. We tried to set rules to sort these by using a tool called Named Entity Recognizer (NER) made by Stanford University [16] and WordNet [17]. When the sentences or the phrases are passed on to NER, it gives in return all the recognized named entities appeared in input texts. WordNet is a lexical database for English, where all words are interlinked to each other in a conceptual semantic ways. By using these tools, we will transform nodes into ontology objects.

- Text values in each node are processed through Named Entity Recognizer.
- If the text value in a node is recognized by NER, then it is considered as individuals and is separated from the hierarchical tree but if it is not recognized by NER, it is considered as class and stays in the tree.
- When the same text values are found in other nodes, then it is considered as entities or classes. These named nodes maintain the hierarchical relationships with other tags linked by <*rdfs:subClassOf*>.
- Nodes separated from the tree need to be classified into higher-concept. Thus, we use WordNet hypernyms to recognize the common parent concept. The conceptualized terms are considered as classes. The separated nodes are considered as individuals and are mapped to the conceptualized class node by <*rdf:type*>.

The conversion of the HTML *List* tags into skeleton ontology is done through the procedures above. The methods we used up to this point are the HTML refinement approach mentioned in section 2. However, with only using this single approach, the information extracted is limited to the small amount of texts that is only used for navigation. More information can be extracted from HTML documents which we try to get through sentence analysis with some rules applied to it. The detailed procedures will be followed in the next subsections.

3.2.2 Extracting sentences from <*p*> tags: To add more details onto the skeleton, we extract texts from <p> tags categorized as *Paragraph* in Table 1, which generally contain main text of HTML document. However, it is unrealistic to go through all the texts within the tags and classify them into categories and recognize their relationships. To reduce the excessiveness of the information, we tried to narrow our dataset by picking the sentences that contain named entities recognized by NER. We tried to recognize all the named entities and extract the sentences that contain these nouns. The following illustrates the procedures:

- Use Jsoup API to discard unnecessary elements and extract only the texts located inside <*p*> tags.
- Each sentence is distinguished by period '.'.
- All the sentences are processed through NER. If there is a named entity in the sentence recognized by the tool, then this sentence is added to our dataset. Also, the origin of the sentences is extracted from title tag to keep track of where it is from.

3.2.3 Retrieving triples: With the set of extracted sentences, it is essential to define *subject*, *predicate* and *object* based on the English syntax to map it on to the skeleton ontology. Thus, we used Open Information Extractor (OIE) to get the triples for each sentence. It is a tool made by Stanford Natural Language Processing group that analyzes English sentences and in return gives back all possible sets of triples for each sentence [16]. All the sentences extracted from the previous stage are processed through OIE. When the sentence is passed on to the tool, it splits sentences into clauses for it to be able to analyze the sentences. In return we get a set of all possible triples for each sentence.

For the returned set of triples, we check if the title value of the document where the triples were extracted from is equal to the text value located in previously extracted nodes. If there is a match, we map *subject* and *object* as individuals and *predicate* as an object properties by <*owl:ObjectProperty*> onto OWL 2. Extracted *subject* and *object* are linked to the extracted origin values by <*rdf:type*>. In the case where there are more than one triple set returned, various *object* candidates possible for one *subject*, then all the triples are treated as the same individuals which are grouped with <*owl:sameAs*>.

3.2.4 Manipulation through ontology editor: Basic ontology is established by going through previous procedures. However, the accuracy of the modules proposed is heavily dependent on the quality of the tools used which in return might give the wrong or missing triples. For example, some of the information could be left out due to named entities unrecognizable by NER. Therefore, more configuration of the established ontology is needed to make the full ontology based on the purpose of its own use. We used Protégé [23], OWL 2 editor made by Stanford.

4. BUILDING ONTOLOGY ON GLOMIS DOMAIN

In this section, we will demonstrate how our modules are implemented in the domain website. As an example, we chose the domain GLOMIS [19], which contains the information regarding European and Korean joint-degree program. In the subsection 4.1, there will be an explanation of the domain GLOMIS. In the subsections 4.2 and 4.3, the procedures described in the sections from 3.1 to 3.2 will be explained.

4.1 Domain GLOMIS

Website GLOMIS is not built in HTML5 but in the formal version of template-driven HTML. It contains the 'Site-Map' which is formed with the sequences of *List* (<*ul*>, <*li*>) tags that act exactly as same as the tags contained in <*nav*> in HTML5. We used these sequences of *List* tags to build the skeleton ontology which contains information of the relationships among HTML documents in the domain.

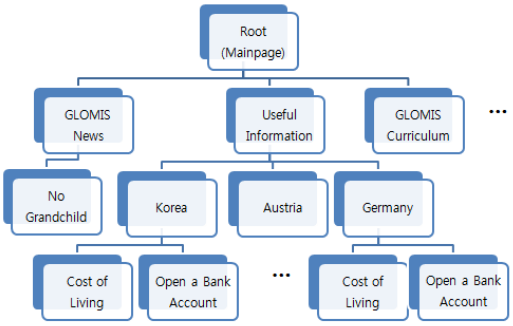


Fig. 2. Visualized sample of tree built from GLOMIS domain

Assigning the main-page of the GLOMIS website as root node, it has a total of 74 nodes, also meaning 74 different URLs within the domain. Some of the nodes are represented in Fig. 2. Every node placed underneath other nodes which are connected with lines represents the hierarchical relationship between the nodes. For instance, node named *Useful Information* placed underneath *Root* is a child-node of *Root* and it contains three child-nodes which are *Korea, Austria*, and *Germany*.

```
<ul>
 <li> <a href="/board_tQCd06" class="strong ">GLOMIS News</a> </li>
 <li> <a href="/page_DYiT33" class="strong ">GLOMIS Curriculum</a> </li>
 <ul>
  <li> <a href="/page_CCKj57">Pai Chai University</a> </li>
  <ul>
   <li> <a href="/page_CCKj57">Curriculum</a> </li>
   <li> <a href="/page_XKBI08">Application</a> </li>
   <li> <a href="/page_qmEP58">Registration</a> </li>
   <li> <a href="/page_YtUM52">Grants</a> </li>
   <li> <a href="/page_ywSV92">Contact Person</a> </li>
  </ul>
  <li> <a href="/page_NSeX66">Chungbuk University</a> </li>
  <ul>
   <li> <a href="/page_NSeX66">Curriculum</a> </li>
   <li> <a href="/page_cPgw17">Application</a> </li>
   <li> <a href="/page_osNM32">Registration</a> </li>
   <li> <a href="/page_XzoJ52">Grants</a> </li>
   <li> <a href="/page_eyCZ67">Contact Person</a> </li>
  </ul>
  <li> <a href="/page_uRtk96">Hildesheim University</a> </li>
  <ul>
   <li> <a href="/page_uRtk96">Curriculum</a> </li>
   <li> <a href="/page_OdNO06">Application</a> </li>
   <li> <a href="/page_fBNX25">Registration</a> </li>
   <li> <a href="/page_wrPL43">Grants</a> </li>
   <li> <a href="/page_QwUp68">Contact Person</a> </li>
  </ul>
  <li> <a href="/page_DgAq23">Karl-Franzens-University Graz</a> </li>
  <ul>
   <li> <a href="/page_DgAq23">Curriculum</a> </li>
   <li> <a href="/page_rImH00">Application</a> </li>
```

Fig. 3. Portion of *List* tags in 'site map'

4.2 Building hierarchical tree

The HTML document for the main page of domain GLOMIS is passed onto the parser made with Jsoup API and additional method to make the hierarchical tree with the values located in the nested tags. With Jsoup, we selected <*ul*> tags and all the tags that are placed within the boundary of the first <*ul*> tag. In return, we get the pure lists of <*ul*> and <*li*> tags that constitutes the whole architecture of the domain. Fig. 3 shows the sample of the pure sequences of *List* tags.

We then extracted the child-nodes of the root. These nodes are the values in <*li*> tags within the boundary of the first <*ul*> tags and the tree depth of one. In return we get *GLOMIS News, Useful Information, GLOMIS Curriculum* and *Student Life*. Note that each node contains its own values as defined in 3.1.1. For example, node *Useful Information* contains URL: "*page_Oaes18*", its own text: "*Useful Information*" and parent-node of root.

The next step for tree construction is to get all the child-nodes of each node *GLOMIS News, Useful Information, GLOMIS Curriculum* and *Student Life*. These nodes are the <*li*> tags placed within the boundary of second <*ul*> tags and at the tree depth of two. However, we first needed to find out whether each of them contains child-nodes or not. To do so, we checked the sequence of *List* tags. For instance, *GLOMIS News* is followed by the tag <*li*> and *Useful Information* is followed by the tag <*ul*> in the sequence of *List* tags. *GLOMIS News* is interpreted as not having any set of *List* tags, therefore no existence of child-nodes. However, for *Useful Information*, it is interpreted as having a set of *List* tags, so all the <*li*> placed within the boundary of <*ul*> which are placed at the tree depth of two becomes a child-node of node *Useful Information*. These are the nodes with text value of *Korea, Austria* and *Germany.*

The same rule applies in order to get the last child-nodes placed at the tree depth of three. For instance, the child-nodes of *Korea* are *Cost of Living* and *Open a Bank Account*.

4.3 Mapping onto ontology and adding details

All the text values of the nodes are processed through the tool NER to transform them into OWL syntax. If the text value is recognized by this tool, it is separated from the hierarchical tree and it is processed through the tool WordNet to find a common concept. For instance, the names of countries, *Korea, Austria, and Germany* are noticed by the NER so it is separated from the hierarchical tree. These are then processed through WordNet to find the common concept which in this case is *Country*. The concept *Country* is mapped as an entity and each nation, *Korea, Austria, Germany* is mapped as individuals linked to the *Country* by <*rdf:type*>.

When the same text values are found in other nodes then it is considered as entities and maintains the hierarchical relationships with the parent or the grandparent nodes. In the case of *Cost of Living* and *Open a Bank Account* which is placed under each nation nodes, it is occurring more than once within the tree. Thus, it is mapped as entities and keeps the hierarchical relationship with *Useful Information* which is the grandparent node because the parent node *Korea, Austria, Germany* was separated in the previous step. The example of skeleton of an ontology extracted from the tags constituting web architecture is shown in Fig. 4.

```
<SubClassOf>
    <Class IRI="#Cost_of_Living"/>
    <Class IRI="#Useful_Information"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Useful_Information"/>
    <Class IRI="#GLOMIS"/>
</SubClassOf>
<ClassAssertion>
    <Class IRI="#Country"/>
    <NamedIndividual IRI="#Austria"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Country"/>
    <NamedIndividual IRI="#Germany"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Country"/>
    <NamedIndividual IRI="#Korea"/>
</ClassAssertion>
```

Fig. 4. Sample of skeleton in OWL/XML syntax

```
GLOMIS  is      European-Korean
GLOMIS  is      European-Korean Joint Degree Program
GLOMIS  is      Joint Degree Program
```

Fig. 5. Triple samples by OIE

In order to add more details onto the skeleton of ontology, sentences containing named entities are extracted from <*p*> tags and are converted into triples. These triples are mapped to the skeleton ontology based on the English syntax analyzed by OIE. For example, sentence *'GLOMIS is a European-Korean Joint Degree Program'* is extracted from the main page, since the word *GLOMIS* is noticed by NER. Now by using the OIE, we were able to get three possible sets of triples for one *subject GLOMIS* as shown in Fig. 5.

All three triples are mapped to an already established entity *GLOMIS* on the skeleton ontology by <*rdf:type*>. Three *objects* in the returned set, *European-Korean, European-Korean Joint Degree Program, Joint Degree Program*, are considered as same individuals by <*owl:sameAs*> and are mapped to *GLOMIS* by <*rdf:type*> as shown in Fig. 6.

```
<ClassAssertion>
    <Class IRI="#GLOMIS"/>
    <NamedIndividual IRI="#European-Korean"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#GLOMIS"/>
    <NamedIndividual IRI="#European-Korean_Joint_Degree_Program"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Country"/>
    <NamedIndividual IRI="#Germany"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#GLOMIS"/>
    <NamedIndividual IRI="#Joint_Degree_Program"/>
</ClassAssertion>
<SameIndividual>
    <NamedIndividual IRI="#European-Korean"/>
    <NamedIndividual IRI="#European-Korean_Joint_Degree_Program"/>
    <NamedIndividual IRI="#Joint_Degree_Program"/>
</SameIndividual>
```

Fig. 6. Mapping triples to OWL 2

Going through the procedures, basic ontology is constructed. From this point, we loaded the established ontology onto Protégé to fix wrong relations and to add more information that was left out throughout the process onto the ontology. For example, all the names of courses and modules which is placed under children node *GLOMIS Curriculum* in Fig. 2 were not recognized by NER. This led to a misconstruction of ontology without any information regarding curriculum of GLOMIS.

5. CONCLUSION

Despite the fact that HTML documents contain lots of information, most of the works on ontology construction was based on the fundamental technologies recommended by W3C. To go through all the semantic layers depicted by W3C, the process of construction is time consuming and difficult even for an expert of the domain. Thus, we have proposed a module that reuses information in HTML documents to construct a basic ontology in a semi-automatic way. To do so, we have implemented a mixture of structure refinement and linguistic approaches. For the structural refinement approach, we have extracted elements contained in sequences of *List* tags, made them into a tree hierarchical structure based on the specified rules, and mapped onto the ontology skeleton. These were the tasks done mainly to put relationship information among the HTML documents in the same domain into an ontology which were not tried in the prior works. As for the linguistic approach, we have extracted all the sentences placed within the <*p*> tags filtered by the NER, processed through IOE to get triples, and mapped triples onto the established ontology skeleton. These were the tasks done mainly to overcome the problems happening when using only the structural approaches.

Our work will help the people who do not know much about ontology construction. Though our work has achieved its own goal of reusing HTML documents and simplifying the process of ontology construction, to make it as a fully automated system and to improve the quality there are more researches to be conducted in the future. The biggest problem in our approach is that some of the information that should have been included on the ontology was left out mainly due to the unrecognized named entities by NER. This might be solved using machine learning technologies or improving the quality of NER. However, as it is well known in the natural language processing field, it is difficult to include all the comprehensive number of cases for the recognition of named entities. Thus, more study on the linguistic approach as well as the alternatives should be conducted in the future for general HTML to OWL 2 conversions. Another problem with the method introduced in this paper is that it is tested only on the domain GLOMIS. Method which is applicable in a general HTML document is therefore needed.

REFERENCES

[1] Thomas R. GRUBER, "Toward principles for the design of ontologies used for knowledge sharing?," International journal of human-computer studies, vol. 43, issue. 5, 1995, pp. 907-928.

[2] Basic Formal Ontology, Overview, 2014. *Online, http://infomis.uni-saarland.de/bfo/overview* - [Last accessed Jul. 14, 2015]

[3] Suggested Upper Merged Ontology, Home, 2015. *Online*, *http://www.adampease.org/OP/index.html* - [Last accessed Jul. 14, 2015]

[4] Hyoun-Soo KWAK, Su-Kayoung Kim, Yeong-Geun Kim, and Kee-Hong Ann, "A Conversion System of HTML Document into OWL Ontology Language, Korean journal Information Processing Society, vol. 11, no. 2, 2004, pp. 539-542.

[5] Taimao SUN, Yiyeon YOON, Wooju KIM, "A Conversion from HTML5 to OWL Ontology," Journal of Society for e-Business Studies, vol. 18, no. 3, 2013.

[6] Jsoup: Java HTML Parser, 2015. *Online*, *http://jsoup.org/* - [Last accessed Aug. 25, 2015]

[7] TOUTANOVA, Kristina, et al., "Feature-rich part-of-speech tagging with a cyclic dependency network," In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, Association for Computational Linguistics, 2003, pp. 173-180.

[8] Luciano DEL CORRO and Rainer GEMULLA, "Clausie: clause-based open information extraction," In: Proceedings of the 22nd international conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2013, pp. 355-366.

[9] Gabor ANGELI, Melvin Johnson PREMKUMAR, and Christopher D. MANNING, "Leveraging Linguistic Structure for Open Domain Information Extraction," In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL, 2015, pp. 26-31.

[10] Hoon HWANGBO and Hongchul LEE, "Reusing of information constructed in HTML documents: A conversion of HTML into OWL," In: Control, Automation and Systems, ICCAS 2008, International Conference on. IEEE, 2008, pp. 871-875.

[11] Uta PRISS, "Formal concept analysis in information science," Arist, vol. 40, no. 1, 2006, pp. 521-543.

[12] Lauren WOOD, et al. *Document Object Model (DOM) Level 3 Core Specification*, 2000.

[13] Saikat MUKHERJEE, et al., "Automatic discovery of semantic structures in html documents," In: Proceedings of the Seventh International Conference on Document Analysis and Recognition-Volume 1, IEEE Computer Society, 2003, p. 245.

[14] Min-Gu Kim, "An Intelligent Taxonomy Relation Extraction System for Automatic Ontology Construction," Ph.D. Thesis, Ajou University, Suwon, Republic of Korea, p. 105.

[15] Bernardo Cuenca GRAU, et al, "OWL 2: The next step for OWL," Web Semantics: science, services and agents on the World Wide Web, vol. 6, no. 4, 2008, pp. 309-322.

[16] The Stanford Natural Language Processing Group: Software, 2014. *Online, http://nlp.stanford.edu/software/index.shtml* - [Last accessed Jul. 22, 2015].

[17] George A. MILLER, "WordNet: a lexical database for English," Communications of the ACM, vol. 38, no. 11, 1995, pp. 39-41.

[18] Universal Dependencies, Universal dependency relations, 2014. *Online, http://universaldependencies.github.io/docs/#language-u* - [Last accessed Aug. 5, 2015].

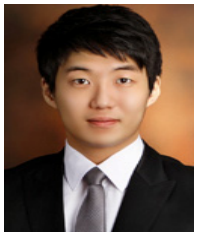
[19] GLOMIS, What is GLOMIS?, 2014. *Online, http://glomis.pcu.ac.kr/* - [Last accessed August 18, 2015].

[20] David NADEAU and Satoshi SEKINE, "A survey of named entity recognition and classification," Lingvisticae Investigationes, vol. 30, no. 1, 2007, pp. 3-26.

[21] The Stanford Natural Language Processing Group, Stanford Named Entity Recognizer, 2015. *Online, http://nlp.stanford.edu/software/CRF-NER.html* - [Last accessed Feb. 15, 2016].

[22] The Stanford Natural Language Processing Group, Stanford Open Information Extraction, 2015. *Online, http://nlp.stanford.edu/software/openie.html* - [Last accessed Feb. 15, 2016].

[23] Protégé, Products, 2015. *Online, http://protege.stanford.edu/support.php* - [Last accessed Feb. 15, 2016].



Chan jong Im
He received the B.S. in International Business from PaiChai University, Korea in 2014. He is currently in Hildesheim-University, Germany, as a master's student. His main research interests include information retrieval, data mining, and machine learning.

Do wan Kim
He received the B.A., M.A. and Ph.D. in Informatics from the University of Regensburg, Germany. He was senior researcher in ETRI. Since 1997 he is professor at PaiChai University. He has interested in semantic web technologies, artificial intelligence, software quality evaluation and software ergonomics.