Using Combined List Hierarchy and Headings of HTML Documents for Learning Domain-Specific Ontology

Muhammad Ahsan Raza[1]
Department of Information Technology
Bahauddin Zakariya University
Multan, Pakistan

Binish Raza[2]
Faculty of Computer Science and Information Technology
University of Malaya
Kuala Lumpur
Malaysia

Taiba Jabeen[3]
Faculty of Education
Allama Iqbal Open University, Multan, Pakistan

Sehrish Raza[4]
Institute of Computer Science and Information Technology
The Women University, Multan, Pakistan

Munnawar Abbas[5]
Department of Computer Science
Institute of Southern Punjab, Multan, Pakistan

***Abstract*—HTML pages contain unstructured and diverse information. However, these documents lack semantics and are not machine understandable. Semantic webs aim to add formal semantics to web data, whereas ontology provides formal semantics to a domain and is thus considered a foundation of semantic webs. Domain ontologies can be constructed manually, but this process is tedious and inefficient. Thus, this study presents an ontology learning (OL) model to create domain ontologies automatically from a set of HTML pages. The key insight of this research is that it combines the list structure and headings of HTML pages to recognize the ontology vocabulary. The approach also incorporates synonym relationships with ontology and allows the semantic interpretation of ontology concepts. We implement the proposed OL approach to build sports ontology from a collection of sports domain HTML documents. The new sports ontology is tested using FaCT++ reasoner; results show no inconsistency in the ontology. Furthermore, experts evaluate the successful mapping of HTML lists and headings to the ontology vocabulary. The proposed OL approach performs effectively and achieves 92.7% and 95.4% precision values for list and heading mapping, respectively.**

Keywords—Ontology learning; semantic web; sports ontology; HTML documents; knowledge extraction; ontology engineering

I. Introduction

HTML is a markup language that is used to write web pages over the World Wide Web [1]. It consists of elements called tags, which have a fixed definition. Web browsers are tools that interpret these tags and display the web pages. Many web applications, such as data mining, machine learning, artificial intelligence, and natural language processing, facilitate the retrieval of information from web pages to fulfill user information requirements [2-4]. However, semantics (i.e., definition of data embedded in a tag) are not explicitly provided in HTML pages. The vision of semantic webs is to achieve HTML documents that are understandable by machines. To achieve this vision, a formal manner of representing semantics is required. This semantic representation organizes information, thereby enabling the machines to search and process information rapidly and accurately. Ontology has emerged as an approach that represents the machine-understandable semantics of a domain and is currently considered the heart of semantic web technologies [5].

An ontology represents domain semantics in terms of classes, which are linked via relationships called properties. The manual construction of ontologies for specific domains is a time-consuming and tedious task [6]. In contrast to manual ontology development, ontology learning (OL) aims to create ontologies automatically from given sources, such as textual and HTML documents or relational database (RDB) schema [7]. Thus, an OL approach helps reduce the time and effort consumed in ontology development. In [8], the authors presented and analyzed a broad spectrum of OL approaches.

This study presents an OL approach that learns ontology automatically by using HTML documents. The ontology is learned through a combined use of the list and heading tags of HTML. In our OL approach, an initial ontology is initially built by exploiting the structure of HTML lists. Subsequently, the HTML headings are mapped and merged into the initial ontology to generate the final ontology. The proposed OL approach has two unique features, which are as follows.

1) It utilizes the combination of HTML list and heading tags to develop an ontology.

2) Synonym relationships are added to the resultant ontology to improve the semantic interpretation of concepts.

The next section presents related work on OL and illustrates the different categories of OL approaches. Section 3 discusses the steps and algorithms of the proposed OL approach. Section 4 outlines the evaluation metrics and discusses the results. The last section provides the conclusion and future directions.

II. Related Work

Various OL techniques have been proposed in the literature. These techniques are classified into three main categories, namely, textual, knowledge, and semistructured based techniques.

A. *Textual-based OL Techniques*

Textual or linguistic techniques depend on natural language processing methods for learning ontology constructs from textual data. These approaches exploit linguistic analysis to uncover the key terms and relationships among terms from a given text. Authors in [9] exploited the syntactic patterns of sentences to discover the dependency relations among words. Their proposed extraction procedure provides a fruitful tool for learning domain ontology to support web services. Two different evaluations, namely, quantitative and qualitative evaluation, are adopted to check the performance of tools. In quantitative evaluation, the precision measure is calculated to represent the extraction of relevant information from the text. On the contrary, the extraction of valid hierarchical structures to build an ontology among words is analyzed through qualitative evaluation.

Venu et al. [10] illustrated relation pattern hypernymy (i.e., parent–child) and meronyms (i.e., part–whole) in their system to learn ontology automatically. The proposed system developed an ontology in five stages, as follows: (1) in the first stage, an iterative focused crawler is used over the corpus collection; (2) in the second stage, the dominant terms are extracted using a hyperlink-induced topic search algorithm [11]; (3) in the third stage, hypernym and meronym patterns are extracted to recognize taxonomic relations (superclass and subclass); (4) association rules are used for mining nontaxonomic relations in the fourth stage; (5) the last stage refines the domain-specific ontology. Many other techniques, namely, co-occurrence analysis [12], clustering analysis [13], term subsumption [14], and association rule mining [15], are also used in the OL procedure for ontology building with high-level precision.

B. *Knowledge-based OL Techniques*

In this OL category, ontologies are learned through structured data, such as using knowledge structure or database schemas. Various approaches have been proposed for learning ontologies from relational schemas, that is, mapping relational schema elements to ontology vocabulary [16-17]. In [18], the authors proposed a migration approach that generates resource description framework (RDF) graphs from the RDB. To build ontologies, they built a prototype that extracts the metadata schema of databases. Subsequently, the extracted schema was converted into a canonical data model to facilitate the migration procedure. Lastly, the structure of the RDF ontology was generated as a result of the migration process.

To learn the ontology from the RDB, Hazber et al. [19] proposed a novel approach to facilitate semantic web applications. The approach consists of two phases, namely, (1) constructing ontology structures from the RDB schema and (2) learning ontology instances from the RDB data accordingly, that is, mapping rules are applied on the RDB data to obtain ontological instances in RDF triple formats. The resultant ontology of the proposed approach appeared to be reliable and was also verified by software engineers. Gamallo and Pereira-Farina [20] used WordNet knowledge structure for OL. Different WordNet relation types, such as synset and hypernyms, are exploited to learn the vocabulary of ontology. The learned ontology represents high-level domain semantics with the use of WordNet knowledge-based techniques.

C. *Semistructured-based Techniques*

The approaches that learn ontologies from semistructured data, such as XML documents or HTML corpus, fall under the semistructured-based OL group. In [21], the ontology in RDF language is learned from legal XML documents. In addition to XML files, authors explored the use of cases of an ongoing project to improve the accuracy of RDF graphs. The performance evaluation of RDF ontologies showed improvement over an existing parser.

Algosaibi and Albahli [22] reviewed different categories of OL techniques, especially focusing on web documents to achieve the vision of semantic webs. Hazman et al. [23] presented an approach to learn ontology from HTML documents. The approaches are used by extracting the phrases of HTML headings, which are then converted into seed concepts representing the domain knowledge. Subsequently, the relationships between heading phrases are identified on the basis of the heading hierarchy within the HTML page. The approach provides a useful lightweight ontology. However, focusing only on HTML headings may not fully capture the semantics of the domain of interest. Authors in [24] followed a manual approach to construct ontologies in the tourism domain. The researchers collected tourism data from HTML datasets and then explored the structure of HTML documents to extract the ontology vocabulary. The tourism ontology is evaluated by experts using questionnaires and the Pellet reasoner tool. The approach is effective but relies on the manual identification of ontology vocabulary from the corpus.

Recent research has focused on automatic learning of ontology from the HTML documents. However, this study proposed an OL technique that differs from existing works in terms of two unique facets. (1) In addition to HTML headings, we used ordered and unordered lists, given that these HTML lists can be a good source to extract web documents with appropriate structure. (2) We included synonym relationships in the creation of final ontologies to improve the semantics of the domain.

III. Proposed Ontology Learning Model

We propose an OL model using semistructured web data while considering all semantics of the domain. The proposed model (as shown in Fig. 1) is composed of seven components.

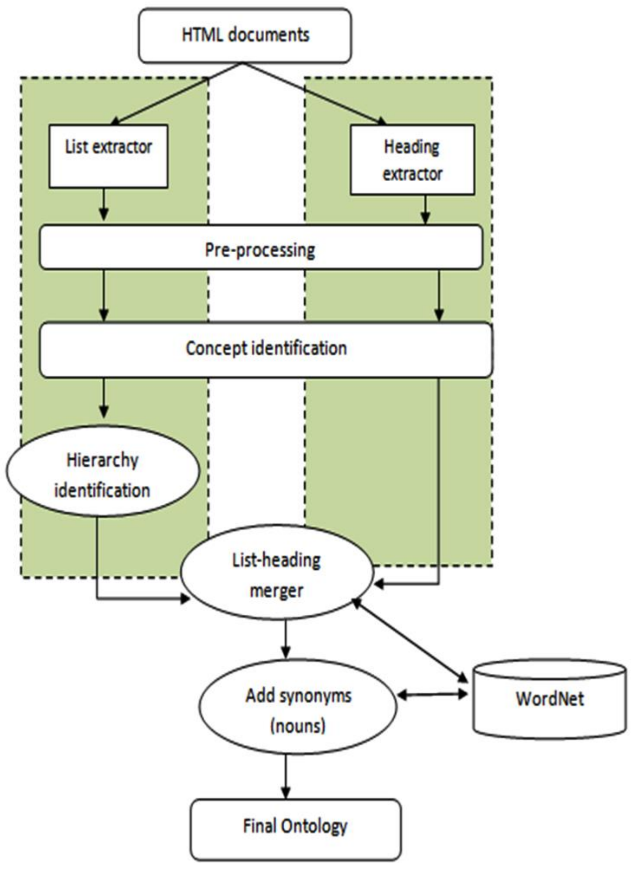


Fig. 1. Steps of the OL Model.

A. List Extractor

HTML lists (either ordered or unordered) within the <script> or <div> tags are usually used as menus in HTML documents [25]. For OL, the lists provide an improved source of information to understand the appropriate structure of the collected web documents. A list extractor module is responsible for extracting lists from the input HTML collection. For instance, from the unordered list (as shown in Fig. 2), the list extractor module acquires <li> athletic trainer </li> as a list item.

B. Heading Extractor

Headings in HTML documents are also important to understand the structure of web documents. Researchers have focused on using them in OL procedures [22-23]. In our OL model, the heading extractor module was used to extract headings from HTML documents, where all six heading levels are considered. For instance, from the HTML code <div><h1> Assistant athletic trainer </h1></div>, this module retrieves the heading item as <h1> Assistant athletic trainer </h1>.

C. Preprocessing

This step normalizes the inner text of the extracted HTML lists and headings. The output of this model are the following two sets: HTML list set (HLS) and HTML heading set (HHS). HLS or HHS normalization involves two important tasks, namely, stop word removal and stemming.

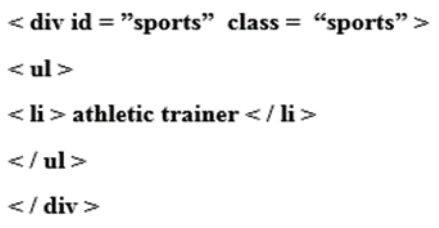
1) Stop word removal: Numbers and stop words (e.g., the, an, to, and of) in the HLS and HHS are removed using this module. For example, from the HHS heading <h2> the athletic trainer </h2>, the stop word "The" is removed from the heading, and the remaining heading "Athletic trainer" is the output.

2) Stemming: The process of reducing different grammatical forms of a term to its base form is called stemming. Primitive stemmers work on the removal of prefixes or suffixes from the text. Various stemmers have been used in the literature [26] to obtain the basic concepts of a domain. We used Porter stemmer [27] for this task.

The overall work of the preprocessing module is shown as an algorithm in Fig. 3.

D. Concept Identification

In this step, two sets of terms (HLS and HHS), which are obtained as output of the preprocessing algorithm, are recognized as concepts by adding an underscore between different terms of a phrase. For instance, a phrase "athletic trainer" in HLS becomes "athletic_trainer," and "associate athletic trainer" is converted to "associate_athletic_trainer." The same procedure is applied to HHS terms. The concepts are then used to identify different relationships to create an ontology



```
< div id = "sports"  class =  "sports" >
< ul >
< li > athletic trainer < / li >
< / ul >
< / div >
```

Fig. 2. Snippet of HTML Document.

Algorithm: HTML Preprocessor

Input: HTML lists and HTML headings

```
1. For each tag // <h1> to <h6> || <ol> || <ul>
2. Extract inner text T of <li> or heading tag
3. Remove noise from T
4. Stem T using Porter stemmer
5. End For
```

Output: Two sets of refined terms: HLS and HHS

Fig. 3. Preprocessing Algorithm.

E. Hierarchy Identification

The hierarchy module builds a parent–child or is–a relation (hierarchy) between the identified HLS concepts. To represent the work of this module, we proposed an algorithm as shown in Fig. 4. The algorithm uses HLS concepts as input, generates subclass relations between concepts by calling the Insert_ontology() function, and finally builds an initial ontology from the HLS.

F. List and Heading Merger

We also focused on the HTML heading list for final ontology construction. To this end, we merged the initial ontology (developed from HLS) with HHS. Fig. 5 describes the detailed steps of merging via an algorithm. The algorithm matches the headings within the HHS with each concept of initial_ontology (OC). If a heading is exactly matched with the OC, then this heading is merged with the OC (e.g., lines 3 and 4). If the heading concept is unmatched with the OC, then its parent is explored. If a parent match is found for OC, then the heading concept is placed under the parent concept (see algorithm lines 7–10). On the contrary, if no match is found, then the heading is inserted as a child of a superclass "root" (e.g., line 13).

1) Similarity measure: An important substep of the HHS merging algorithm is to find a similarity match between concepts (e.g., lines 3 and 9 in Fig. 5). To this end, Wu and Palmer's (WP) measure [28] is used to compute the similarity match among the concepts. For instance, Table I indicates the WP similarity values (calculated via 1) between initial ontology concept (c1) and heading concept (c2).

$$sim_{wp}(c_1, c_2) = \frac{2*depth(lso(c_1,c_2))}{len(c_1,c_2)+2*depth(lso(c_1,c_2))} \quad (1)$$

```
Algorithm: List to ontology convertor
Input: Script S, ol tags, ul tags
1.  For each ol | ul in S
2.      Extract inner text T
3.      If ol | ul is in Div-id && Div-class then
4.      Extract first level <li> inner text T
5.      //insert T under root ontology concept
        CALL Insert_ontology (T, root)
6.      End If
7.      If nested <ol> | <ul> then
        Extract nested level <li> inner text T`
8.      //insert T` under non-root ontology concept T
        CALL insert_ontology (T',T)
9.      End If
10. End For
Output: Initial_ontology
```

Fig. 4. Algorithm for Converting an HTML List to an Ontology.

```
Algorithm: Heading merger
Input: HHS, Initial_ontology
1   For each heading term Ti in HHS
2    Match Ti with initial_ontology concept (OC)
3    If Ti is exactly match with OC
4    Merge Ti in OC
5    End If
6    Else
7    Extract Ti parent Tp
8    Match Tp with OC
9    If Tp match found with OC
10    Insert Ti under Tp
11   End If
12   Else
13    Insert Ti under concept root
14   End Else
15   End Else
16  End For
Output: Final_ontology
```

Fig. 5. Algorithm for Merging Headings.

TABLE I. WU AND PALMER SCORES

C1 (ontology)	C2 (HHS)	WP value
Athletic_trainer	Robust_trainer	0.987
Associate_athletic_director	Associate_ robust _director	0.981
Sports_ physician	Sports_psychologist	0.51
Medical_assistant	Physical_education_instructor	0.30

G. Add Synonyms

The last step of our OL model adds synonyms to the new ontology as additional semantic data. The synonyms for each ontology concept are derived using WordNet knowledge structure and are inserted via sim–syn relationships. We only focused on noun synonyms to be inserted in our ontology via sim–syn relationship because most concepts in the final ontology are nouns. Fig. 6 describes an algorithm for this module that identifies and inserts synonyms in the final ontology concepts.

```
Algorithm: Synonym adder
Input: Final_ontology, WordNet
1   For each concept Ci in Final_ontology
2    Retrieve synonyms S using WordNet for Ci
3    Extract Noun synonyms N from S
4    Insert N using sim-syn relationships with Ci
5   End For
Output: Final_ontology with synonyms
```

Fig. 6. Adding Synonym Algorithm.

IV. Implementation and Results

The proposed OL algorithms (as listed in Section 3) are implemented in Java environment to build a system prototype. In addition, Jena (an open source java API) is used for ontology manipulation, and Protégé tool is used to view the final ontology. We evaluated our approach by using the sports domain dataset, which consists of 105 HTML documents collected from https://www.sports.ru website[1]. Fig. 7 represents the ontograph view (i.e., protégé tool plugin) of the final ontology in the sports domain, which is semantically learned via the system prototype.

A. Evaluation Measures

Two performance measures, namely, semantic reasoner and precision measure, are used to evaluate the performance of our proposed OL model.

1) Semantic reasoned: Once an OL technique learns an ontology, the consistency of new ontology vocabulary should be checked. Semantic reasoners are tools that assess the consistency (duplicate classes or properties and unconnected taxonomy) of ontologies. Different types of semantic reasoners, such as FaCT++, RACER, and HermiT, are available to check the validity of ontologies [29].

2) Precision: Precision evaluation metrics were used to measure the performance of our proposed OL algorithms. This metric shows whether a relevant ontology vocabulary is retrieved by the system prototype from the HTML document set. Precision can be calculated using (2).

$$Precision = \frac{\{relevant\ items\} \cap \{retrived\ items\}}{\{retrived\ items\}} \quad (2)$$

B. Result Analysis

We initially evaluated the new ontology learned by our OL model by using a semantic reasoner. We used the FaCT++ reasoner to check the consistency of our ontology. In our system, the FaCT++ reasoner provided consistent and reliable results for the newly learned sports ontology. This finding indicates that the resultant ontology is consistent and no ambiguity, and redundancy is found in the vocabulary of ontology.

We further evaluated our approach by commissioning experts, which are divided into two groups. Each expert group consists of domain researchers and master students with background in computer science. The experts manually calculated the list items and headings from the collected web pages of the sports domain, where 70 list items spanning three levels of list hierarchy and 1080 heading items are extracted. Then, the experts compared these items with the sports ontology vocabulary that is learned by our OL model. To elaborate the expert's results, the precision value was computed using Eq. 2 (as discussed in previous section). The two precision values calculated are (1) for list items that are used to build the initial ontology and (2) for headings that are mapped to create the final ontology. Fig. 8 provides a graphical representation of the precision value showing 92.7% precision for list mapping and 95.4% for heading mapping. This finding suggests that our OL approach can stably learn the ontology from the combined use of the lists and headings of HTML documents.

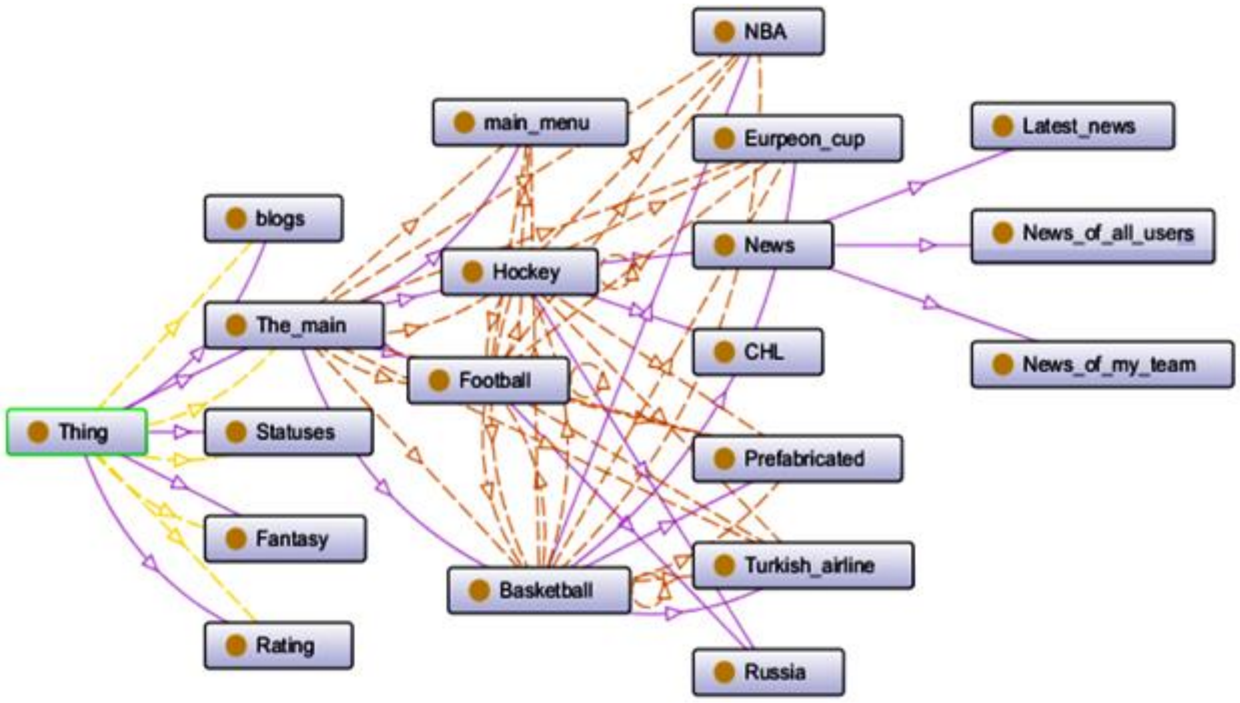


Fig. 7. Ontograph view of Sports Ontology.

[1]https://www.sports.ru//

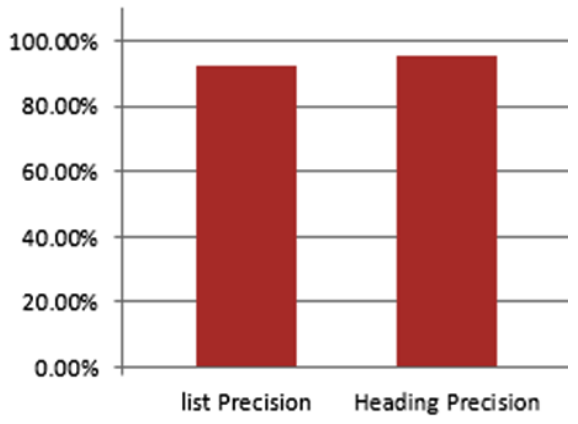


Fig. 8. List and Heading Precision.

V. Conclusion

This research addresses the issue of accurate OL from the HTML corpus by focusing on the combined use of the lists and headings of HTML documents. We propose an OL model and claim that this model can accurately map the HTML dataset to an ontology knowledge base. We have tested our OL prototype over an HTML corpus in a sports domain. The ontology learned from the web dataset using our approach shows 100% consistency on the semantic reasoner. Our OL approach obtains 92.7% and 95.4% precision for HTML list mapping and HTML heading mapping, respectively. This finding indicates that the combination of HTML lists and headings is useful in learning precise ontology vocabulary from HTML documents.

In the future, we will focus on improving our OL algorithms for inferring HTML heading hierarchy along with the HTML lists from the web documents. Furthermore, we will attempt to utilize synonym (sim–syn) relationships of newly learned ontologies for inferring the accurate structure of HTML headings.

References

[1] D. Raggett, A. L. Hors, and I. Jacobs. "HTML 4.01 Specification W3C Recommendation 24 December 1999," 20 January 2020; https://www.w3.org/TR/html401/.

[2] V. N. Gudivada, D. L. Rao, and A. R. Gudivada, "Chapter 11 - Information Retrieval: Concepts, Models, and Systems," Handbook of Statistics, V. N. Gudivada and C. R. Rao, eds., pp. 331-401: Elsevier, 2018.

[3] L. E. M. FERNÁNDEZ, and S. Bhulai, "Recommendation System for Netflix," Master, Vrije Universiteit Amsterdam, 2018.

[4] L. C. Smith, "Artificial Intelligence in Information Retrieval: forty years on," The Human Position in an Artificial World: Creativity, Ethics and AI in Knowledge Organization: ISKO UK Sixth Biennial Conference London, 15-16th July 2019, D. Haynes and J. Vernau, eds., pp. 301-302, Baden-Baden: Ergon-Verlag, 2019.

[5] M. A. Raza, M. Rahmah, A. Noraziah, and M. Ashraf, "Sensual Semantic Analysis for Effective Query Expansion," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 9, no. 12, 2018.

[6] G. Li, J. Lin, J. Lu, and Q. Xu, "Extraction of Ontological Terminology Relations of Scheduling Regulations Based on Combination Method," in 2019 International Conference on Electronic Engineering and Informatics (EEI), 2019, pp. 370-374.

[7] B. Sathiya, and T. V. Geetha, "Automatic Ontology Learning from Multiple Knowledge Sources of Text," International Journal of Intelligent Information Technologies (IJIIT), vol. 14, no. 2, pp. 1-21, 2018.

[8] R. Lourdusamy, and S. Abraham, "A Survey on Methods of Ontology Learning from Text," Intelligent Computing Paradigm and Cutting-edge Technologies. pp. 113-123, 2020.

[9] M. Sabou, C. Wroe, C. Goble, and G. Mishne, "Learning Domain Ontologies for Web Service Descriptions: An Experiment in Bioinformatics," in Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, 2005, pp. 190-198.

[10] S. H. Venu, V. Mohan, K. Urkalan, and G. T.V., "Unsupervised Domain Ontology Learning from Text," in Mining Intelligence and Knowledge Exploration MIKE 2016, Springer, Cham, 2017, pp. 132-143.

[11] Y. Du, X. Tian, W. Liu, M. Wang, W. Song, Y. Fan, and X. Wang, "A novel page ranking algorithm based on triadic closure and hyperlink-induced topic search," Intelligent Data Analysis, vol. 19, pp. 1131-1149, 2015.

[12] S. Suresu, and M. Elamparithi, "Probabilistic relational concept extraction in ontology learning," Int. J. Inform. Technol., vol. 2, no. 6, 2016.

[13] Z. Xu, M. Harzallah, and F. Guillet, "Comparing of Term Clustering Frameworks for Modular Ontology Learning," Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management pp. 128-135, 2018.

[14] H. N. Fotzo, and P. Gallinari, "Learning « generalization/specialization » relations between concepts: application for automatically building thematic document hierarchies," in Coupling approaches, coupling media and coupling languages for information retrieval, Vaucluse, France, 2004, pp. 143–155.

[15] Z. Abedjan, and F. Naumann, "Improving RDF Data Through Association Rule Mining," Datenbank-Spektrum, vol. 13, no. 2, pp. 111-120, 2013/07/01, 2013.

[16] L. Man, D. Xiao-Yong, and W. Shan, "Learning ontology from relational database," in 2005 International Conference on Machine Learning and Cybernetics, 2005, pp. 3410-3415 Vol. 6.

[17] M. A. Raza, and B. Raza, "Comparative Analysis of Ontology Extraction Techniques from Relational Database," Science International, no. 4, pp. 3589-3595, 28-4-2016, 2016.

[18] H. Ling, and S. Zhou, "Mapping Relational Databases into OWL Ontology," International Journal of Engineering and Technology, vol. 5, pp. 4735-4740, 2013.

[19] M. A. G. Hazber, R. Li, Y. Zhang, and G. Xu, "An Approach for Mapping Relational Database into Ontology," in 2015 12th Web Information System and Application Conference (WISA), 2015, pp. 120-125.

[20] P. Gamallo, and M. ı. Pereira-Farina, "Compositional Semantics using Feature-Based Models from WordNet," in Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications, Valencia, Spain, 2017, pp. 1–11.

[21] A. Crotti Junior, F. Orlandi, D. O'Sullivan, C. Dirschl, and Q. Reul, "Using Mapping Languages for Building Legal Knowledge Graphs from XML Files," arXiv e-prints; 2019. https://ui.adsabs.harvard.edu/abs /2019arXiv191107673C.

[22] A. Algosaibi, and S. Albahli, "Web Documents Structures as Source for Machine-Understandable Document," in Proceedings of the 2019 2nd International Conference on Intelligent Science and Technology, Durham, United Kingdom, 2019, pp. 11–17.

[23] M. Hazman, S. El-Beltagy, and A. Rafea, "Ontology Learning from Web Organization Documents," International Journal for Metadata Semantics and Ontologies, vol. 4, pp. 24-33, 01/01, 2009.

[24] C. Chantrapornchai, and C. Choksuchat, "Ontology construction and application in practice case study of health tourism in Thailand," SpringerPlus, vol. 5, no. 1, pp. 2106, 2016/12/20, 2016.

[25] S. Langridge, DHTML Utopia: Modern Web Design : Using JavaScript & DOM, 1st ed., Collingwood Australia: Sitepoint Publishing, 2005.

[26] A. G. Jivani, "A Comparative Study of Stemming Algorithms," International Journal of Computer Technology and Applications, vol. 2, no. 6, pp. 1930-1938, 2011.

[27] M. Porter. "Snowball: A language for stemming algorithms," 20 January 2020; http://snowball.tartarus.org/texts/introduction.

[28] L. Meng, R. Huang, and J. Gu, “A review of semantic similarity measures in wordnet,” International Journal of Hybrid Information Technology, vol. 6, no. 1, pp. 1-12, 2013.

[29] D. Tsarkov, and I. Horrocks, “FaCT++ Description Logic Reasoner: System Description,” in Automated Reasoning, Springer, Berlin, Heidelberg, 2006, pp. 292-297.