Indian Journal of Science and Technology, *Vol 8(16), 60439, July 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Framework for Generation of RDF Data from HTML

Arup Sarkar[1], Subha Koley[1], Ujjal Marjit[2*] and Utpal Biswas[1]

[1]Department of Computer Science & Engineering, University of Kalyani, Kalyani - 741235, West Bengal, India; tellarup@gmail.com, utpal01in@yahoo.com
[2]C.I.R.M. University of Kalyani, Kalyani - 741235, West Bengal, India; marjitujjal@gmail.com

Abstract

Generally, legacy data of various domains have been expressed through different formats over the web. However, most of the contents are available in static html format. Nowadays, the majority of the current web pages are designed and developed in dynamic HTML. Different kind of advanced templates have been used. Until now, there is a plethora of web pages available in static HTML format to describe through the RDF format. In this article, an Ontology based framework has been proposed to nurture the static HTML pages and exploit them in RDF format for backup purpose as well as better reusability in the domain of Semantic Web.

Keywords: HTML, Ontology, RDF, Regular Expression, Semantic Web

1. Introduction

A new era of digital age has started to flourish the way of information sharing and processing. From the very beginning, a numerous number of programmers, scientists, developers have paid a lot of attention to enhance the World Wide Web (WWW). The inventor of World Wide Web, Sir Tim Berners-Lee came up with newer concepts for the World Wide Web with the promises of new possibilities. He brings a new concept called Semantic Web[1,2]; a web of data[1,3] along with their meanings. It makes us to believe that data could be machine processable as well as machine understandable. Here, a new data representation model such as Resource Description Framework (RDF)[4] in Semantic Web domain has been introduced to describe the data more expressively through its semantics. Now, the information is expressed in machine processable and understandable way through the use of RDF technology.

In general, semantic gives meaningful information about the data to understand the hidden meanings. Thereafter, any description regarding a living or non-living entity is denoted as semantic data. However, this explanation of semantic data leads to a simpler view regarding the term, while describing anything semantically is possible to become very complex as well as complicated in nature. Semantic web deals with such semantic descriptions about web information. Under this circumstance, it is required to understand why we need such semantic description about those documents scattered over the web. First of all, it is needed to know how the traditional web actually works. Traditional web is nothing but web of documents i.e. a huge storage of information. This information normally as web documents. The collection of documents is called 'web' because most of these documents are somehow connected with each other through hyper-links. The links among the documents are visualized as spider web made by web documents. Hence, it is called web of documents. Machines as well as different web applications are intended to access and process the web documents. Sometimes, they mean to represent these documents over the web browser in a specific format as per the instruction given in those documents through syntaxes. Machines or applications mechanically process them followed by the syntaxes and executes as per the

*Author for correspondence

predefined instructions. This behaviour of machines or applications about the web documents explain one thing that all the information which is held in those documents are machine processable, but they cannot understand it as humans do. Meaningful semantics are added to make the data available from these web documents. Normally the semantic descriptions are generated separately using RDF statements. Different and more than one ontologies are used to describe the terms semantically. These kind of semantic descriptions are saved as documents, they also known as semantic mark-ups. Generation of semantic mark-ups is taken here as the first step. Next step is to add a link to the original document so that any software agent traversing through that web document may know that a semantic description for the page is present on the web for further processing.

2. Literature Survey

According to the existing literature in the domain of semantic web, publishing legacy data of different domain into RDF format is not a new attempt. Many works already have done in this research area. In this article, a framework has been proposed to convert the contents of static HTML page to RDF. A few numbers of works are already presented in this domain. The main limitations of the existing works are, their design is for specific kind of web projects only. That means, if the target web pages under consideration are changed, then the whole configuration for the working framework has to be changed. Although, the implementation of the framework has been reconsidered. There is no doubt to accept that these kinds of jobs are pretty much hectic and annoying. Though, there are lots of differences present in various approaches.

During the last few years, extraction of structured data from web pages is an important concern in this field. Many similar works have been proposed throughout the literatures. Arasu et al.[5] proposed a methodology to extract structured data from web pages. The authors clearly explained the methods of structured data extraction. In their method, they proposed some rules to automatically deduce the template followed by a page or a collection of pages without any human generated input. The algorithm automatically finds out unknown template which is used by the target pages and subsequently extracts information from the pages. The algorithm is known as 'ExAlg' that works fine against some data rich sites, wherein every page followed a pattern or template to gather information throughout the pages.

A new approach called Data Rover has been proposed by Davulcu et al.[6] A taxonomy based crawler has been applied in the system to select and extract data from websites. Data Rover takes the page's DOM (Document Object Model) tree as input to apply a page segmentation algorithm to find out the different logical segments of the pages. The authors have been defined a page segmentation algorithm to identify different sections from the DOM tree of the page of a product. They also used a website as input that describes about the products. But, it is important that all the content has been categorized properly to ensure the success of their works. However, they did not mention how the proposed system will be applied to other kinds of web pages where the contents are not well categorized. In our view, it seems to be a limitation, since every kind of static HTML pages are not well structured. However, it will perform well in a particular domain of websites.

Cafarella et al.[7,8] described an aggregated approach of different types of methodology. The authors[7] showed how three different projects such as 'TextRunner', 'WebTables' and 'Deep-web crawling' have been combined to extract structured data from different types of web page. These promising approaches are about to extract structured data from web. More precisely, the natural language processing approach in the TextRunner project may prove to be useful to solve our problem. Our point of interest centre around the TextRunner, because only this work targets the normal text over the web while other two, i.e. WebTables and Deep-web crawling focus on the data available from the HTML tables and backend databases used on the web respectively. These tables and databases are quite structured in nature as well as much easier to extract than extracting structured data from a piece of natural unstructured text.

Myllymaki[9] discussed regarding a XML based software framework called ANDES to extract semi-structured data from the target websites to make them better structured.

A cluster based approach 'ClustVX' has been illustrated by Grigalis[10] to retrieve structured data from the web. The main drawback of this project is the inability to deal with the complete unstructured data.

One of the most popular projects about structured data extraction is the DBpedia project[11]. The target of DBpedia is to extract the structured data of page con-

tents or articles from Wikipedia[12]. DBpedia performs the aforesaid job through the framework called DBpedia Information Extraction Frameowrk (DIEF)[13]. It also uses ontology to describe the key terms and contents extracted from the Wikipedia. The main drawback of this project is, its framework is designed considering highly categorized and template based data. More specifically, it is designed to extract structured data from Wikipedia pages and publish them as Linked Data over the web of data by interlinking them with different knowledge bases.

Dbpedia-Live[14,15] has been launched with same approach that follows a live synchronization method, so that all the current updates within Wikipedia immediately get reflected inside the DBpedia-Live dataset.

3. Brief Overview on Structured Data, Unstructured Data and Semantic Data

It does not matter if the data is structured or unstructured, since both types of data represent some information. However, most of the time, structured data are possible to chop down into different homogeneous categories. Unstructured data are not possible to treat alike the structured data due to its heterogeneous nature.

As of now, semantic data represents some sort of structured data along with their meaning. Semantic data are collection of meaningful structured data.

Structured data means, the data is possible to break down into chunks of data. The meaningful pieces of data, as like as key and value pairs. At the beginning this key-value pairs may not be very meaningful for further use since the pairs come as a collection of keyed terms that can be uniquely identifiable. These terms also come without any information about the relations among each other. Addition of more information describing their meaning will help to understand the use of these terms, but it is a challenging prospect. Here, the semantic web technology plays a key role.

Unstructured data represents such information, which is not decomposable easily into key-value manner. It is a very difficult job to identify the key information from the unstructured data and it is almost impossible to add semantics to them.

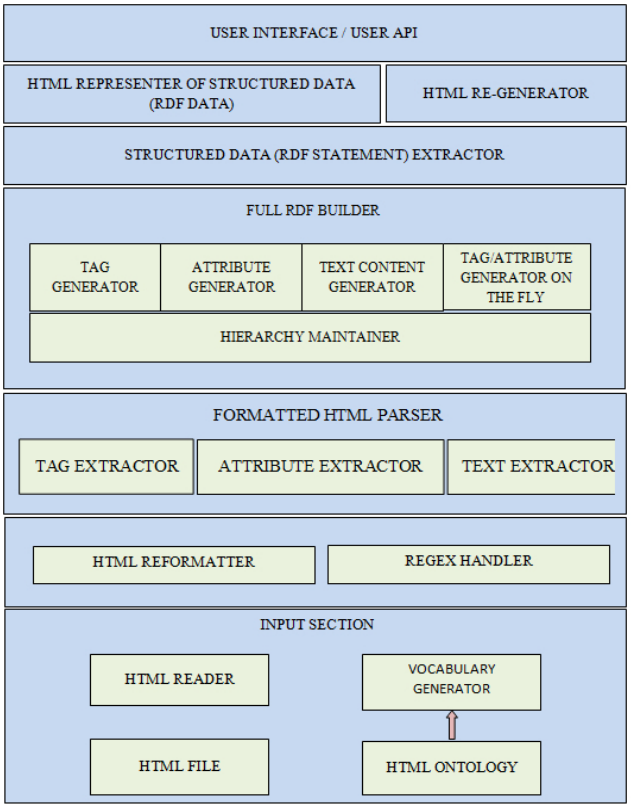

Figure 1. Schematic Diagram of the Proposed Framework.

4. The Proposed Ontology Based Framework

The proposed framework is divided into eight modules, i) Input Section, ii) Input Handler, iii) Formatted HTML Parser, iv) Modified RDF Builder, v) Structured Data Extractor, vi) HTML Representer of Structured Data, vii) HTML Regenerator and finally viii) User Interface. Before understanding the core of the system, it is required to clear the idea about Modified RDF, RDF and HTML in the context of the proposed framework. A schematic diagram of the framework is shown in Figure 1.

The following framework has been designed based on ontologies[16,17]. Two ontologies have been developed for the experiment. The excerpt of the statements from both ontology files are shown in Figure 2 and Figure 3.

4.1 Modified RDF

In the context of the proposed framework, Modified RDF represents the RDF (Resource Description Framework)

```
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/H1">
    <rdfs:label xml:lang="en">h1</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/H2">
    <rdfs:label xml:lang="en">h2</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/H3">
    <rdfs:label xml:lang="en">h3</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/H4">
    <rdfs:label xml:lang="en">h4</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/H5">
    <rdfs:label xml:lang="en">h5</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/H6">
    <rdfs:label xml:lang="en">h6</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Head">
    <rdfs:label xml:lang="en">head</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Header">
    <rdfs:label xml:lang="en">header</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/NormalTags"/>
  </owl:Class>
```

```
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Background">
    <rdfs:label xml:lang="en">attr_background</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Bgcolor">
    <rdfs:label xml:lang="en">attr_bgcolor</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Border">
    <rdfs:label xml:lang="en">attr_border</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Cellpadding">
    <rdfs:label xml:lang="en">attr_cellpadding</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Cellspacing">
    <rdfs:label xml:lang="en">attr_cellspacing</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Char">
    <rdfs:label xml:lang="en">attr_char</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Charoff">
    <rdfs:label xml:lang="en">attr_charoff</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology/Attr_Charset">
    <rdfs:label xml:lang="en">attr_charset</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology/Attributes"/>
  </owl:Class>
```

Figure 2. (a) Code Snippets from HTML Ontology of Elements, (b) Code Snippets from HTML Ontology of Attributes.

```
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/Html">
    <rdfs:label xml:lang="en">HTML</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Document"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/Navigation">
    <rdfs:label xml:lang="en">navigation</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Document"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/Panel">
    <rdfs:label xml:lang="en">panel</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Html"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/Paragraph">
    <rdfs:label xml:lang="en">paragraph</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Html"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/Section">
    <rdfs:label xml:lang="en">section</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Html"/>
  </owl:Class>
-<owl:Class rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/Heading">
    <rdfs:label xml:lang="en">heading</rdfs:label>
    <rdfs:subClassOf rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Document"/>
  </owl:Class>
```

```
-<owl:DatatypeProperty rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/preamble">
    <rdfs:label xml:lang="en">preamble</rdfs:label>
  -<rdfs:comment xml:lang="en">
      Describes the abstarct or preamble of the document...
    </rdfs:comment>
    <rdfs:domain rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
-<owl:DatatypeProperty rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/keyInfo">
    <rdfs:label xml:lang="en">key information</rdfs:label>
  -<rdfs:comment xml:lang="en">
      Describes the key information collected from certain tags like blink, marquee etc.
    </rdfs:comment>
    <rdfs:domain rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
-<owl:DatatypeProperty rdf:about="http://www.tellarup.com/ontologies/HtmlOntology_struct/hasHeading">
    <rdfs:label xml:lang="en">has heading</rdfs:label>
    <rdfs:comment xml:lang="en">Has a heading within the body tag</rdfs:comment>
    <rdfs:domain rdf:resource="http://www.tellarup.com/ontologies/HtmlOntology_struct/Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
```

Figure 3. (a) Code Snippets from HTML Ontology_struct of Elements, (b) Code Snippets from HTML Ontology_struct of Attributes.

statements to describe the syntactical information of a dataset, such as a static HTML file. Generally, RDF has been used to describe the raw data semantically. These semantic descriptions are the collection of RDF statements.

Normally, those RDF statements represent only the key information. Most of the time, the semantic web developers and researchers are concerned only about this key information or structured data. In the context of the static HTML pages, the meaning of structured data is exactly the same, that is, the key information from which further information is deduced. However, an HTML page not only holds the data to display but it also holds some syntactical information which tells how that information should be displayed. This syntactical data normally get ignored during the processing of HTML pages to extract the key information and to represent them in RDF format. It is possible to represent this RDF data in HTML representation but it will never look same as before. It is impossible to get back the original HTML representation just like the originating source. In this article, Modified RDF is introduced to bridge the gap. Basically, Modified RDF is the RDF representation of complete HTML page including all information, even the syntax related description.

The hierarchy of the nested HTML tags, their attributes, texts between the tags and also their order of occurrences, every bit of information is preserved. Modified RDF is described as the backup of original source HTML file using RDF statements. Although the use of Modified RDF is limited due to the presence of huge number of blank nodes[18]. Too much blank nodes incur difficulties during the execution of 'SPARQL' queries, since complexity of the RDF statements highly increased. Reduction of this unwanted complexity is a challenging task. However, if it is reduced and SPARQL queries start running smoothly, then only use of Modified RDF solves both the problems of backup of HTML pages as well as extracting meaningful key information from the HTML pages on the fly.

4.1.1 Problems Related to Blank Nodes

Blank nodes[19,20] do not really represent any node within a RDF graph. However, this kind of assumption does not reflect the complete truth about the blank nodes. Actually blank nodes represent 'things' within a RDF graph which may exist without an identity or name. Besides, blank nodes are still required to refer inside the graph to maintain the consistency i.e. the proper SPO (Subject-Predicate-Object) structure of all the RDF statements. As mentioned before they don't possess a proper identity or name that is why their processing is difficult while squeezing further information.

Blank nodes help to make a RDF graph consistent while also makes it complicated for processing. Huge number of blank nodes within a graph grows the complexity of SPARQL queries. In addition to that, they may get failed to execute successfully. Sometimes, presence of blank nodes becomes a problem while parsing. Due to this reason, it is preferred to avoid the blank nodes as much as possible. However, it seems quite useful and difficult to avoid their use.

4.2 The Proposed HTML-to-RDF (H2R) and RDF-to-HTML (R2H) Framework

The proposed framework has been divided into multiple modules and sub-modules. The following description about the framework has been started from the bottom-to-top. The first module at the bottom is called the 'input section'. This module consists of four components which represent the initial elements to continue the processing further through the framework. They are usual target 'HTML File' and 'HTML Reader' to read the entire HTML content out of it. Next one is the original HTML ontology that already mentioned in the section of Modified RDF. The fourth component of this module is the 'Vocabulary Generator'. This Vocabulary Generator is important in terms of programming, since it helps to maintain the actual list of RDF elements or vocabulary which will use to describe the different parts of a HTML page.

4.2.1 Input Section

At the outset, this module performs the initiation process of the system. It is comprised of the following four components i) HTML File, ii) HTML Reader, iii) HTML Ontology and iv) Vocabulary Generator.

The first component i.e. HTML File stands for the primary input to the system. This HTML page has to be used to generate the Modified RDF. In addition, this file is used to extract the structured data.

The HTML Reader reads the code from HTML page or retrieves from the location of the HTML page over the web. It holds all the data into the memory as an input for processing in the next module, i.e. 'Input Handler'.

```
[rbr0]<html>[rbr0]
[rbr1]<head>[rbr1]
[rbr2]<title>[rbr2][rft0:rbr2]Link Test[rft0:rbr2][rbr3]</title>[rbr3][rbr4]</head>[rbr4]
[rbr5]<body bgcolor = "black" text="gold">[rbr5]
[rft1:rbr5]I have something to say:[rft1:rbr5]
[rbr6]<h1>[rbr6][rft2:rbr6]Hello World![rft2:rbr6][rbr7]</h1>[rbr7][rbr8]<br />[rbr8]
[rft3:rbr8]How are you?[rft3:rbr8][rbr9]<br />[rbr9]
[rbr10]<h2>[rbr10][rft4:rbr10]Follows are some of my favorite links:[rft4:rbr10][rbr11]</h2>[rbr11]
[rbr12]<ol>[rbr12]
[rbr13]<li>[rbr13][rft5:rbr13]Favorite movie DB[rft5:rbr13][rbr14]</li>[rbr14]
[rbr15]<ul>[rbr15]
[rbr16]<li>[rbr16][rbr17]<a href="http://www.linkedmdb.com">[rbr17]
[rft6:rbr17]LinkedMDB[rft6:rbr17][rbr18]</a>[rbr18][rbr19]</li>[rbr19]
[rbr20]</ul>[rbr20][rbr21]<li>[rbr21]
[rft7:rbr21]Favorite Search engine[rft7:rbr21][rbr22]</li>[rbr22]
[rbr23]<ul>[rbr23]
[rbr24]<li>[rbr24][rbr25]<a href="http://www.google.com">[rbr25]
[rft8:rbr25]Google[rft8:rbr25][rbr26]</a>[rbr26][rbr27]</li>[rbr27]
[rbr28]</ul>[rbr28][rbr29]</ol>[rbr29]
[rbr30]</body>[rbr30]
[rbr31]</html>[rbr31]
```

Figure 4. Sample Code Snippet of Formatted HTML Code.

The Ontologies are represented by the component HTML Ontology. The ontology files help to assist the extraction process of structured data during the presence of different phases of the framework. At present two different ontologies are generated. First ontology 'HTMLOntology.owl' generates the Modified RDF statements from the raw HTML codes. The second ontology 'HTMLOntology_struct.owl' is used to generate the RDF statements to describe the structured content extracted from the HTML file/ Modified RDF. If required, external ontologies may incorporate within the system.

Vocabulary Generator is important in terms of programming the framework to develop the RDF statements. During the generation of RDF statements different concepts and properties are required and these are described within the ontologies. While programming, these ontological terms are difficult to incorporate directly from the ontologies. Vocabulary Generator basically a programming unit that generates a separate vocabulary of references to each ontological term (i.e. concepts and attributes) that helps during the development of RDF statements.

4.2.2 Input Handler

This module is the composition of two components such as 'HTML Reformatter' and 'Regex Handler'. It is not guaranteed that every input (HTML codes) will be in correct format. The correct format inside the framework assures that, every tag is well formatted, correctly opened and closed. Every tag used in this framework is valid. Within this framework, XHTML is preferred for every HTML file. HTML Reformatter is responsible to make every HTML file well formatted, so that this file becomes usable without any problem. Besides, HTML Reformatter performs one more action, and this action is performed over reformatted i.e. the corrected HTML content. In this phase of operation, it actually encapsulates all the tags and text element within some specific and pre-formatted string components so that every tags and every text element from the HTML page can be easily identified. This operation makes the application of Regex Handler easier. An entirely re-formatted HTML source code is shown in the following Figure 4.

The second component of this module is 'Regex Handler'. Although this component is placed in this module, but it has huge potential of re-usability throughout the framework. Regex Handler is completely based on the use of regular expression. Regular expression is quite useful and handy in certain situation while dealing with different types of texts with a regular type of pattern. Generally, Regular expression is proved to be powerful tool of pattern matching for processing texts. In particular, such type of advantage of regular expression has been used within framework through the Regex Handler. It takes a target text as input and a regular expression, to extract a certain portion from the input text according to the pattern explained by the regular expression. On the other hand, Regex Handler acts as an interface for the other active components from various modules.

4.2.3 Formatted HTML Parser

This module is comprised of the following three separate extractors: i) Tag Extractor, ii) Attribute Extractor and iii) Text Extractor. All of these components are highly dependent on the 'Regex Handler'. The Tag Extractor is responsible to extract all the HTML tags used throughout the HTML page excluding any attributes. Extraction of every attributes against every tag is the task assigned to the Attribute Extractor. HTML tags tell the browser how to represent some text within a page or how a web page will overall look like. The main information about the page is normally its text contents. So, it is important to extract all the texts from the page. This is another task

```
- <rdf:RDF>
  - <h2r:HtmlDoc rdf:about="http://localhost:8080/h2r/sample2a.html">
    - <h2r:hasNestedTag>
      - <h2r:Html>
        - <h2r:hasNestedTag>
          - <h2r:Body>
            - <h2r:hasNestedTag>
              - <h2r:Ol>
                - <h2r:hasNestedTag>
                  - <h2r:Ul>
                    - <h2r:hasNestedTag>
                      - <h2r:Li>
                        - <h2r:hasNestedTag>
                          - <h2r:A>
                              <h2r:occOrder>1</h2r:occOrder>
                            - <h2r:hasAttribute>
                              - <h2r:Attr_Href>
                                  <h2r:attributeValue>http://www.google.com</h2r:attributeValue>
                                </h2r:Attr_Href>
                              </h2r:hasAttribute>
                              <h2r:hasFollowingText>Google</h2r:hasFollowingText>
                            </h2r:A>
                          </h2r:hasNestedTag>
                          <h2r:occOrder>1</h2r:occOrder>
                        </h2r:Li>
                      </h2r:hasNestedTag>
                      <h2r:occOrder>4</h2r:occOrder>
                    </h2r:Ul>
                  </h2r:hasNestedTag>
                - <h2r:hasNestedTag>
                  - <h2r:Li>
```

Figure 5. Excerpt of Code from a Modified RDF File.

performed within this module by the component called Text Extractor. All these extraction operations have been performed by some preconfigured or predicted patterns as it is already introduced within the page during the second stage of operation of HTML Reformatter. Now, these specific patterns are matched using regular expression with the help of Regex Handler.

4.2.4 Modified RDF Builder

As mentioned in section 4.1, Modified RDF has been introduced as a novel concept. The Modified RDF is the RDF representation of an entire HTML page without excluding any content from the pages. An efficient RDF vocabulary or ontology is required to represent every tags, attributes, and texts as RDF element. This is the place where the HTML Ontology and the vocabulary generated in module 1 is used.

Modified RDF Builder is divided into five components, such as 'Tag Generator', 'Attribute Generator', 'Text Content Generator', 'Tag/Attribute Generator on the Fly' and finally the 'Hierarchy Maintainer'. Tag Generator generates the RDF statements to represent the HTML tags into the Modified RDF. As similar as Tag Generator, the Attribute Generator generates the RDF statements to

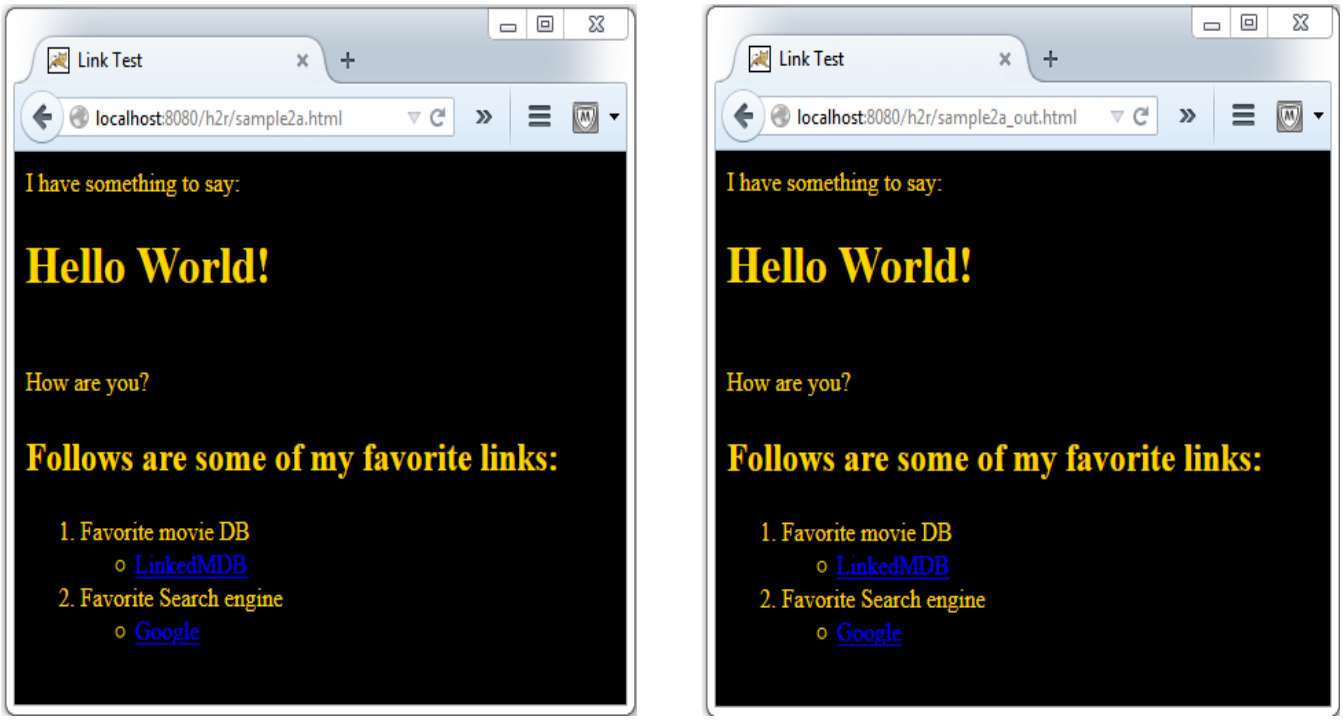

Figure 6. (a) Source HTML File, (b) Regenerated HTML File in Web Browser.

represent the attributes of each HTML tags. In the similar way the Text Content Generator generates the RDF statements to represent the normal texts come after the HTML tags. Every element i.e. tags, their corresponding attributes and the text contents all come in a certain order. If the order gets changed, whole purpose of the Modified RDF generation will be messed up. As a consequence the next modules will not work properly. A complete record of the hierarchy of all the tags, attributes, text contents need to be preserved. The Hierarchy Maintainer does this by contributing the ordering information of every HTML element as RDF statement within the Modified RDF. That is why it becomes easy to get back the original HTML content from the Modified RDF. An excerpt of source code from a Modified RDF file is shown in the following Figure 5.

4.2.5 Structured Data Extractor

'Structured Data Extractor' extracts the actual structured data from the HTML pages as input. This module is the second most important module after Modified RDF Builder. Additionally, it is used to extract main key information from the web pages, so that RDF statements can be generated to make this information available to exploit by the other existing semantic web applications as well as for end user's view.

4.2.6 HTML Representer of Structured Data

This module acquires the RDF data as input and subsequently generates HTML code to represent the RDF data as key-and-value pair format. It shows the RDF data in user friendly HTML format so that any end user be able to see the extracted RDF data as simple HTML page through the web browser.

4.2.7 HTML Regenerator

Modified RDF acts as a back-up of the original HTML page with the help of 'HTML Regenerator'. HTML Regenerator generates exactly same HTML code as of the original input HTML page has. It accepts only the Modified RDF file or its content as input to execute. An example of original HTML file and its regenerated version are shown in the following Figure 6.

In the above figures the left web browser window displays the original HTML page, whereas the right browser window shows its regenerated version from the Modified RDF file.

4.2.8 User Interface

User Interface has the following advantages:

- Prompt to input the location of target HTML file.
- Option to generate the Modified RDF file only.
- Option to generate the actual key structured data in RDF format.
- Option to display the generated RDF data in raw format using various serializations (i.e. N3, NTRIPLES, Turtle, RDF/XML etc.) or in HTML format.
- Option to display the Modified RDF.
- Option to regenerate the original HTML content by quering Modified RDF as input.

5. Implementation of the Proposed Framework

Java platform has been chosen to develop the proposed framework. Besides, Jena libraries have also been integrated to handle the entire RDF related issues. The whole implementation tasks are performed as follows.

In 'Input Section', the main HTML Ontologies have been coded manually without any help of specific standard tool like Protégé. The RDF statements generated through the framework uses the concepts and properties described within the HTML Ontologies. A Java wrapper class has been designed to generate the vocabulary of those concepts and properties from the above ontologies. The auto-generated vocabulary file is basically a Java program, which has been applied along with the Jena[21] libraries. Besides, the component called 'HTML Reader' has also been implemented to read the HTML content (source code) from a given link. To implement the HTML Reformatter, another well-known Java API called JTidy[22] has been applied. It provides different options to check the consistency of the codes within the input HTML page. It also caters the provision to make the input HTML file more consistent with the HTML/XHTML specifications. Before acquiring any HTML file as input, it is concisely checked either they are already semantically annotated by the use of different semantic web technologies in terms of RDFa, Microformats etc., or not. Normally, pre-annotated HTML pages with semantic mark-ups remain ready for the semantic web. The primary target of the framework is the simple legacy HTML files that still exist on the

web of documents and are needed to expose semantically over the semantic web.

A separate Java class has been developed for the Regex Handler to handle the regular expression related problems. Additionally, Regex Handler reformats the tidied (processed with JTidy) HTML file to add some predefined string as shown in Figure 4.

In addition to above, a parser program has been written to tackle the fully reformatted HTML file. The parser helps to extract all the tags, their corresponding attributes, and all the following texts in a specific order.

Another program has been developed to perform the jobs of Modified RDF Builder, i.e. RDF statement generation for every tag, attribute, and text content including information about their occurrence order in the source HTML file. The program is also capable to generate RDF vocabulary for any tag or attribute which is not described in the ontology file. 'Structured Data (RDF Statement) Extractor' is under experiment. 'HTML Regenerator' has been coded separately. Sixth and eighth module i.e. 'HTML Representer' and 'User Interface/ User API' respectively are not fully implemented.

6. Discussion

As discussed in the implementation section, the ontology for the HTML tags and attributes have been coded manually. However, development of the ontology through advanced tools like Protégé is possible. In spite of that, it is manually coded to reduce the complexity of the ontology and to make it simple. A code snippet of the ontologies are shown in Figure 2 and 3 using RDF/XML notation.

The job of the HTML Reformatter is to tidy up the source HTML code as well as to add some predefined strings to specifically identify the exact start and end position of a tag or attribute. It is also required to identify the position of texts occurs between the tags. It is better to call these predefined strings as codes, because each of them has a specific meaning. These codes are combination of characters and numeric values. The character portion reveals, it around a HTML tag or an attribute of HTML tag or just a text content. The numeric parts notify about their occurrence order. In the re-formatted HTML, two types of codes have been used.

The code generation method uses the following rule:

$$X\,I\,,\ where\ X = rbr,\ \ I \geq 0 \qquad (1)$$

The above given expression is valid to generate the codes for tag and attribute identification only. For normal text content identification following rule has been followed:

$$Y\,I : XJ,\ where\ X = rbr,\ Y = rft,\ I \geq 0,\ J \geq 0 \qquad (2)$$

As shown in the equation (2), the code has two parts, one i.e. YI, at the left side of the " : " and XJ at the right of it. The left part represents the text content and its occurrence order within the hierarchy. The second part expresses about its preceding HTML tag, the text content actually trails after. In the Figure 4, the code [rft0:rbr2] in the line 3 is around the first occurring text content "Link Test" where right part of the code tells its preceding HTML tag is encapsulated within the code [rbr2] which is '<title>'. All these generated codes have been embedded within square brackets.

7. Conclusion and Future Works

The proposed framework deals only with the static HTML pages. It completely excludes the web pages generated by any template or any pages generated through CMS (Content Management System) based website. The framework also considers only those HTML documents as valid input for the system which is not semantically annotated by any means.

The newly proposed concept of Modified RDF needs to be explored in depth. At present it is acting like a backup system for the original HTML documents in RDF format. The main problem is that, the use of Modified RDF for any other purpose is very difficult due to the excessive presence of blank nodes inside the code.

Another important module of the framework is the structured data extractor. The strategy followed here is pretty much pre-mature, but justified due to its complicated and very high heterogeneous nature. It is the most challenging part of the framework, where further research and development can be done. The other type of strategies to extract key information, based on Natural Language Processing (NLP), specifically sentiment analysis of terms, frequency of occurrences of terms may play key role here.

The proposed framework serves as a basic platform to prepare any kind of legacy HTML pages for the semantic web. We have plan in near future to include the CMS based pages for this framework.

8. References

1. Berners-Lee T, Hendler J, Lassila O. The Semantic Web. Scientific American. 2001 May; 29-37.
2. W3C Semantic Web Activity [Internet]. 2013 Jul 19 [cited 2014 Dec 12]. Available from: http://www.w3.org/2001/sw/
3. Yu L. Introduction to the Semantic Web and Semantic Web Services. Boca Raton: Chapman & Hall/CRC; 2007.
4. RDF 1.1 Concepts and Abstract Syntax [Internet]. 2014 Feb 25 [cited 2014 Oct 22]. Available from: http://www.w3.org/TR/rdf11-concepts/
5. Arasu A, Garcia-Molina H. Extracting structured data from web pages. International Conference on Management of Data (SIGMOD '03); 2003; San Diego, California, USA, New York: ACM; 2003. p. 337–48.
6. Davulcu H, Koduri S, Nagarajan S. Datarover: a taxonomy based crawler for automated data extraction from data-intensive websites. Proceedings of the 5th ACM International Workshop on Web Information and Data Management (WIDM '03); New Orleans, Louisiana, USA, New York: ACM; 2003. p. 9–14.
7. Cafarella MJ, Madhavan J, Halevy A. Web-scale extraction of structured data. SIGMOD Record. 2008 Dec; 37(4):55–61.
8. Cafarella MJ, Halevy A, Madhavan J. Structured data on the web. Communications of the ACM. 2011 Feb; 54(2):72–9.
9. Myllymaki J. Effective Web Data Extraction with Standard XML Technologies. Proceedings of the 10th International Conference on World Wide Web (WWW'01); Hong Kong, New York: ACM; 2001. p. 689–96.
10. Grigalis T. Towards automatic structured web data extraction system. In: Albertas AL, Dzemyda CG, Vasilecas O, editors. Local Proceedings and Materials of Doctoral Consortium of the Tenth International Baltic Conference on Databases and Information Systems; 2012 Jul 8-11; Vilnius, Lithuania, Vilnius: Zara; 2012. p. 197–201.
11. Yu L. A developer's guide to the semantic web. Berlin, Heidelberg: Springer; 2011. Chapter 10, DBpedia; p. 379–08.
12. Wikipedia [Internet]. 2015 Jan 15 [updated 2015 Jan 15; cited 2015 Jan 16]. Available from: http://en.wikipedia.org/wiki/Wikipedia
13. The DBpedia Information Extraction Framework [Internet]. 2014 Apr 11 [updated 2014 Apr 11; cited 2014 Dec 15]. Available from: http://dbpedia.org/documentation
14. DBpedia Live [Internet]. 2015 Jan 8 [updated 2015 Jan 8; cited 2015 Jan 8]. Available from: http://wiki.dbpedia.org/DBpediaLive
15. Morsey M, Lehman J, Auer S, Stadler C, Hellman S. DBpedia and the live extraction of structured data from Wikipedia. Program: electronic library and information systems. 2012; 46(2):157–181.
16. Noy NF, Mcguinness DL. Ontology development 101: a guide to creating your first ontology. Knowledge System Laboratory, Stanford University; 2001 Mar. p. 24 Report No: KSL-01-05
17. Gruber TR. Toward principles for the design of ontologies used for knowledge sharing?. International Journal of Human-Computer Studies. 1995; 43(5–6):907–28.
18. Blank nodes [Internet]. 2014 Oct 17 [updated 2014 Oct 17; cited 2014 Oct 22]. Available from: http://en.wikipedia.org/wiki/Blank node
19. Milicic V. Problems of the RDF model: Blank Nodes. 2011 Jul 14 [cited 2014 Dec 15] In: Problems of the RDF model [Internet]. Bew Citnames A blog by Vuk Milicic. Available from: http://milicicvuk.com/blog/2011/07/14/problems-of-the-rdf-model-blank-nodes/
20. Arenas M, Consens M, Mallea A. Revisiting Blank Nodes in RDF to Avoid the Semantic Mismatch with SPARQL. RDF Next Steps Workshop; 2010 Jun 26-27; Palo Alto, CA, USA, Palo Alto: W3C.
21. Getting started with Apache Jena [Internet]. Apache Software Foundation; 2011 [cited 2014 Dec 16]. Available from: https://jena.apache.org/getting_started/index.html
22. JTidy - HTML PARSER AND PRETTY-PRINTER IN JAVA [Internet]. 2014 [cited 2014 Dec 15]. Available from: http://jtidy.sourceforge.net/index.html