Semantic Search in Wiki using HTML5 Microdata for Semantic Annotation

Pabitha P[1], Vignesh Nandha Kumar K R[2], Pandurangan N[2], Vijayakumar R[2] and Rajaram M[3]

[1] Assistant Professor, Dept of Computer Technology, MIT Campus, Anna University Chennai 600 044, Tamilnadu, India.

[2] Student, Computer Science and Engineering, Anna University Chennai 600 044, Tamilnadu, India.

[3] Professor, Anna University of Technology, Tirunelveli

Abstract

Wiki, the collaborative web authoring system makes Web a huge collection of information, as the Wiki pages are authored by anybody all over the world. These Wiki pages, if annotated semantically, will serve as a universal pool of intellectual resources that can be read by machines too. This paper presents an analytical study and implementation of making the Wiki pages semantic by using HTML5 semantic elements and annotating with microdata. And using the semantics the search module is enhanced to provide accurate results.

Keywords: **HTML5, Microdata, Search, Semantics, Annotation, Wiki**

1. Introduction

Wikipedia contains vast amount of information and resources. Though it provides vast amount of information,they can be only understandable only by humans. We can make them machine understandable by including the semantic contents in the wiki engine. Thereby,we can make search efficient and optimization.

2. Literature survey

2.1 Semantic web

The term semantic web coined by Tim Berners-Lee, is not a separate web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

Conventional web contains a large pool of information that is human readable but not interpretable by computers. Semantic web extends it by annotating the web pages with semantic description. This allows computers to retrieve information from the web automatically and to manipulate them.

2.2 Ontology

An ontology is the formal explicit specification of shared conceptualization. A conceptualization refers to an abstract model of some phenomenon in the world that identifies the relevant concepts of that phenomenon. Explicit means that the type of concepts used and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine understandable.

2.3 Wiki

A wiki is a Web-based system that enables collaborative editing of Web pages. The most important properties of wikis are their openness and flexibility. Their openness lets each user participate in content creation, and their flexibility supports different users' working styles without imposing technological constraints. Wikis provide a Web-based text editor with a simple mark-up language to create content and to link easily between pages as well as a versioning system to track content changes and full-text search for querying the wiki pages.

2.4 Semantic wiki

A semantic wiki tries to extend a normal wiki's flexibility to address structured data. To this end, it supports metadata in the form of semantic annotations of the wiki pages themselves, they can and of the link relations between wiki pages. The annotations usually correspond to an ontology that defines the properties that can be associated with different object types.

Semantic Wiki offers:

- a simple formalism for semantically annotating links and wiki articles or other kinds of content.
- a semantic search for querying by not only keyword but also semantic relations between objects and

- possibly an additional automatic or semi-automatic extraction of metadata from wiki articles to simplify the annotation process – for example, by topic(EUprojects)or even indirectly (meeting minutes of EU projects)

3. HTML5

HTML5 is the 5th major revision of the core language of the World Wide Web: the Hypertext Mark-up Language (HTML), initiated and developed mainly by WHATWG (Web Hypertext Applications Technology Working Group).Started with the aim to improve HTML in the area of Web Applications, HTML5 introduces a number of semantic elementswhich include: <section>, <nav>, <article>, <aside>, <hgroup>, <header>, <footer>, <time> and <mark>.

These are some of the tags that have been introduced just to bring semantics in web pages, with no effect on the way it is displayed. They behave much like a grouping element such as as far as displaying them is concerned. This means if an old browser cannot recognize these tags it will handle them much similar to the way a grouping element is handled. The semantic elements tell the browsers and web crawlers clearly the type of content contained within the element. For instance states explicitly that the figures within the element represent a time.

4. Microdata

Apart from the semantic elements HTML5 introduces Microdata – the way of annotating web pages with semantic metadata using just DOM attributes, rather than separate XML documents. Microdata annotates the DOM with scoped name/value pairs from custom vocabularies. Anyone can define a microdata vocabulary and start embedding custom properties in their own web pages. Every microdata vocabulary defines a set of named properties. For example, a Person vocabulary could define properties like name and photo. To include a specific microdata property on your web page, you provide the property name in a specific place. Depending on where you declare the property name, microdata has rules about how to extract the property value. Defining your own microdata vocabulary is easy. First, you need a namespace, which is just a URL. The namespace URL could actually point to a working web page, although that's not strictly required. Let's say I want to create a microdata vocabulary that describes a person. If I own the data- vocabulary.org domain, I'll use the URL http://data-vocabulary.org/Person as the namespace for my microdata vocabulary. That's an easy way to create a globally unique identifier: pick a URL on a domain that you control. In this vocabulary, I need to define some named properties.

Let's start with three basic properties:

• name (your full name)
• photo (a link to a picture of you)
• url (a link to a site associated with you, like a weblog or a Google profile)

Some of these properties are URLs, others are plain text. Each of them lends itself to a natural form of markup, even before you start thinking about microdata or vocabularies or whatnot. Imagine that you have a profile page or an ―about page. Your name is probably marked up as a heading, like an <h1> element. Your photo is probably an <img> element, since you want people to see it. And any URLs associated your profile are probably already marked up as hyperlinks, because you want people to be able to click them. For the sake of discussion, let's say your entire profile is also wrapped in a <section> element to separate it from the rest of the page content. Thus:

```
<section itemscope itemtype= "http://data-
vocabulary.org/Person">
    <div itemprop="title" class="title">              President
     </div>
     <div itemprop="name" class="name">
        Mark Pilgrim
     </div>
</section>
```

The major advantage of Microdata is its interoperability, i.e any RDF representation of an ontology can be mapped to HTML5 microdata.

5. Existing System

MediaWiki is a free software wiki package written in PHP, originally for use on Wikipedia. It is now used by several other projects of the non-profit Wikimedia Foundation and by many other wikis. MediaWiki is an extremely powerful, scalable software and a feature-rich wiki implementation, that uses PHP to process and display data stored in its MySQL database. Pages use MediaWiki's wiki-text format, so that users without knowledge of HTML or CSS can edit them easily.

5.1 MediaWiki Architecture

In the architecture of MediaWiki as shown in Fig.1 the top two layers hardly have anything to do with semantic annotation. The layers of concern are the Logic Layer and the Data Layer; the major part lies in Logic Layer.
The following figure shows the architecture of MediaWiki:

User layer	Web browser	
Network layer	Squid	
	Apache web-server	
Logic layer	MediaWiki's PHP scripts	
	PHP	
Data layer	File system	MySQL Database (program and content)

Fig. 1 Architecture of Mediawiki [12]

Logic Layer: This is the core part of MediaWiki that accomplishes the above said tasks. The PHP scripts of MediaWiki are to be edited to carry out these tasks. The parser module (Fig. 6) is to be enhanced to convert between Wiki and HTML markups. Also the data-vocabulary referred in the pages must be validated and appropriate flags must be set.

Data Layer: The MySQL database layout of Mediawiki is so normalized that adding a new table needs no alterations in any table [13]. The metadata about each page is stored in the page table, whose layout is given in Fig.2

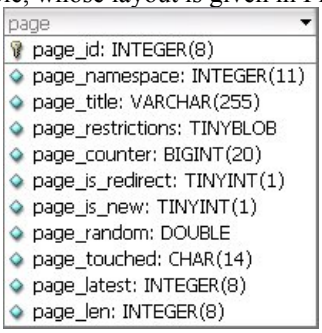

Fig. 2 Layout of the **page** table

The actual content of the page is stored in a separate table named text whose layout is given in Fig.3

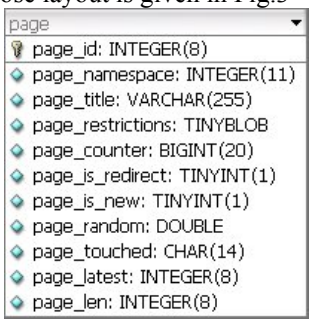

Fig. 3 Layout of the **text** table

Wiki Parser: Here is a sample Wiki markup:

The '''Wikimedia Foundation, Inc.''' is a [[Non-profit organization|nonprofit]] [[Foundation (nonprofit)|charitable organization]] For the Internal Revenue Service (the IRS) to recognize an organization's exemption, the organization must be organized as a trust, a corporation, or an association.

The original HTML syntax markup corresponding to this shown below:

The Wikimedia Foundation, Inc. is a non-profit charitable organization For the Internal Revenue Service (the IRS) to recognize an organization's exemption, the organization must be organized as a trust, a corporation, or an association.

Here, for instance, [[Non-profit organization|non-profit]] corresponds to non-profit. That means the Wiki engine parses the Wiki markup entered by the author and generates the corresponding HTML markup.

6. Proposed system

The Wiki pages, if annotated semantically, will serve as a universal pool of intellectual resources that can be read by machines too.

Mediawiki follows a standard template for its web pages. Thus a search engine or any other software that needs data to be extracted from Wiki pages need not search the entire web page; instead it is enough to search the variable data, i.e. the contents excluding the fixed (template) part [2]. This project is to define a way of annotating the wiki pages using a simple markup similar to that already available for editing conventional wiki pages and to define a set of vocabularies to represent the relationship among Wiki pages. This involves developing a parser to parse the markup and to replace it with actual HTML5 microdata for storing and the vice-versa while editing.

A parser to recognize the Semantic Wiki mark-up and to generate the corresponding HTML5 markup has been developed [1]. Thus the project includes:

- Defining a Wiki mark-up for representing ontology
- Extending the parser for translating this to corresponding HTML5 mark-up
- Defining vocabularies that define entities related to Wiki pages

Enhancing the search engine to take advantage of the

Semantic definitions is being implemented.

To account for the semantic annotations in the pages, we add a new table **microdataobject** whose layout is:

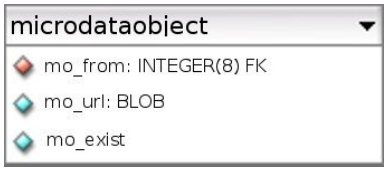

Fig. 4 Layout of the new table **microdataobject**

Block diagram:

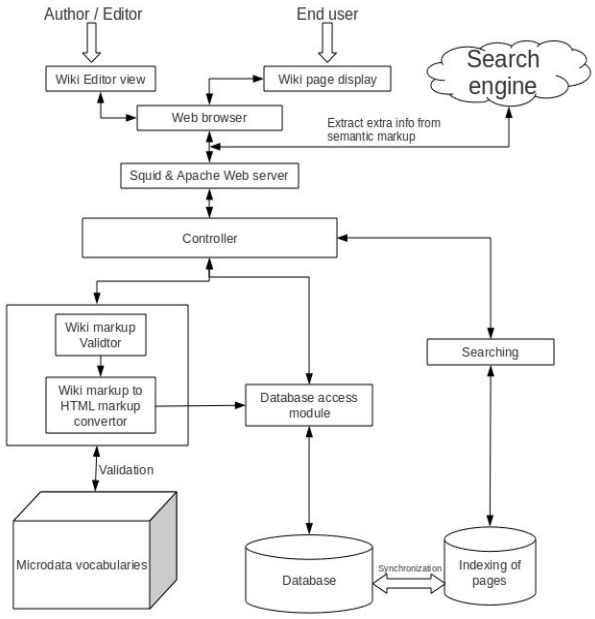

Fig. 5 Block diagram of Semantic Mediawiki

Controller is the module that despatches the requests from the user to the corresponding module. However the Squid (proxy server) may serve the user with cached results from previous requests.

Microdata vocabulary is the actual definition of the class to which the object described in the page belongs to. InHTML5 microdata this is referred to by the value of `itemtype` attribute.

Editor module provides the interface through which a user can edit or create wiki pages. If the user edits an already existing page, the corresponding page is fetched from the database and the HTML markup is converted into wiki markup and is displayed in the editor interface. After the user edits the contents and clicks *Save page* the modified contents are given to the parser to be converted to HTML markup.

6.1 Parser

Parser is the core module that is responsible for validating the wiki markup and converting it to HTML markup to be rendered as a web page. The Wiki markup may be a control markup that does not affect the content of the page – the one which updates the metadata alone, like minor edits. For such cases the parser asks the database access module to update the associated entries in the database.

A part of the newly added modules in Parser.php file:

```
function addSemantics( $text ) {
    wfProfileIn( __METHOD__ );
    $atParaStart = preg_match('/^<p>\{__:/',$text);
    $atParaEnd = preg_match('/__\}<\/p>/',$text);
    $pos = strpos($text,'{__:');
    if($pos == false)
            return $text;
    $pattern = array(
      '/(?<=\{__:)(\w+)/' =>
            'http://data-vocabulary.org/'.'\\1'.'">',
      '/\{__:/' => '<span itemscope itemtype="',
      '/ __\}/' => '</span>',
      '/(?<=@)(\w+)(:")([^"]*)(")/' =>
            '\\1'.'">'.'\\3'.'</span>',
      '/@(?=(\w+))/' => '<span itemprop="' );
    if($atParaStart==1) {
            $text = preg_replace('/^<p>\{__:/','{__:',$text);
            $pattern['/(?<=\{__:)(\w+)/'] = 'http://data-
vocabulary.org/'.'\\1'.'"><p>';
    }
    if($atParaEnd==1) {
            $text = preg_replace('/__\}<\/p>/','__}',$text);
            $pattern['/ __\}/'] = '</p></span>'; } $text = preg_replace(
array_keys($pattern), array_values($pattern), $text);
    wfProfileOut( __METHOD__ );
    return $text; }
```

The wiki markup to include microdata annotation is:

{__:ItemType

… @itempropName:”value” …

__}

For instance, to include microdata annotation about a person, the Wiki markup is as follows:

{__:Person

… @name:"Richard Stallman" …

… @title:"President" …

… @nickname:"RMS"

__}

Here, the ellipsis are used to represent some arbitrary content, just as placeholder; not part of the syntax. This Wiki markup on passing the Parser module becomes:

<span itemscope

itemprop="http://data-vocabulary/Person">

…Richard Stallman…

…President…

…<span itemprop=”"ickname">RMS…

This approach differs from the earlier proposals of semantic wiki using RDF (such as KawaWiki [4] and Rhizome [5]) in simplicity. The user's effort to annotate a web page is reduced drastically as semantic HTML elements and attributes serve the purpose of their XML counterparts. Thus to make the e-resources most updated as well as semantic without much strain HTML5 microdata suits best.

6.2 Mediawiki Search module

The search module of Mediawiki is organised as one base class named SearchEngine and 6 subclasses. SearchUpdate, one of the subclasses, is to update the search index in the database whereas database specific operations are carried out by the other 5 classes, one for each of MySQL, MySQL4, PostgreSQL, SQLite, Oracle and IBM-DB2.

In the base class, some functions are just declared as stub and their actual implmenetation is done in the database-specific subclasses.

The class diagram is as shown below:

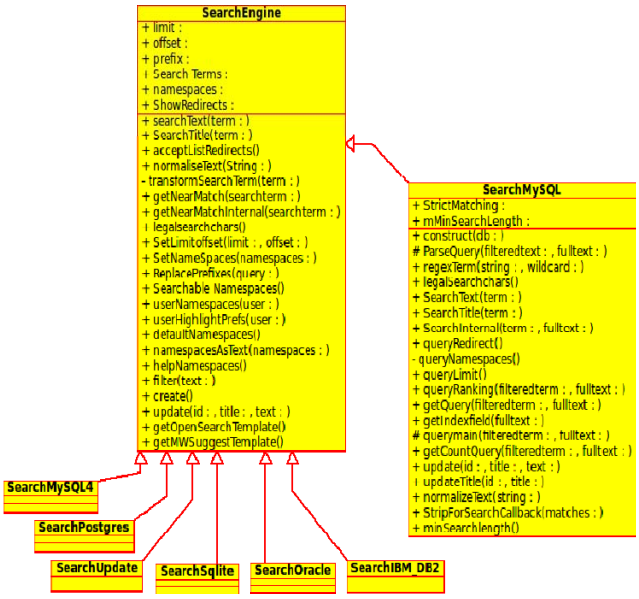

Fig. 6 Class diagram of the search implementation

Flowchart:

The control flow of the search module in Mediawiki is depicted in the following figure. It involves tasks such as preprocessing and normalizing the search text, replacing get arguments with corresponding prefixes, resolving namespaces and so on.

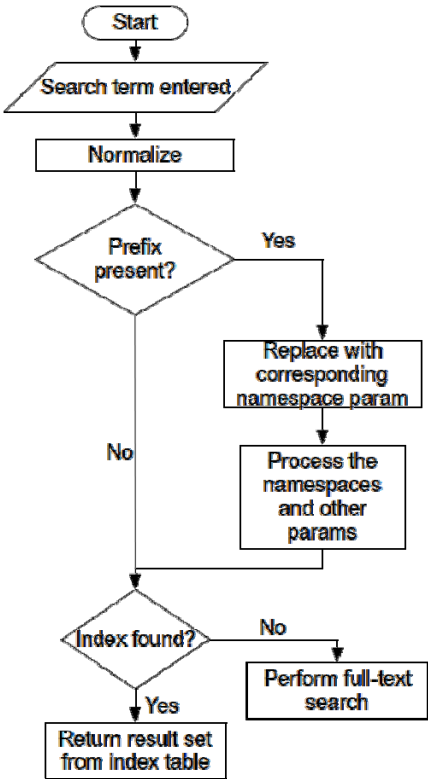

Fig. 7 Flowchart of the search process

A snippet of the search code implemented is as follows:

```
function replacePrefixes( $query ){
  global $wgContLang;
  $parsed = $query;
  if( strpos($query,':') === false )
  { // nothing to do
    wfRunHooks(
'SearchEngineReplacePrefixesComplete', array(
$this, $query, &$parsed ) );
                return $parsed;
  }
  $allkeyword = wfMsgForContent('searchall').":";
  if( strncmp($query, $allkeyword,
strlen($allkeyword)) == 0 ){
    $this->namespaces = null;
    $parsed = substr($query,strlen($allkeyword));
} else if( strpos($query,':') !== false ) {
    $prefix =
substr($query,0,strpos($query,':'));
    $index = $wgContLang->getNsIndex($prefix);
    if($index !== false){
      $this->namespaces = array($index);
      $parsed = substr($query,strlen($prefix)+1);
}
    else {
      $prefix =
substr($query,0,strpos($query,':')-1)
    $parsed = '{_:'.$prefix;
    }   }
  if(trim($parsed) == '')
    $parsed = $query; // prefix was the whole
query
    wfRunHooks(
'SearchEngineReplacePrefixesComplete', array(
$this,
    $query, &$parsed ) );
    return $parsed;
}
public static function userNamespaces( $user ) {
  global $wgSearchEverythingOnlyLoggedIn;
  // get search everything preference, that can
be set to be read for logged-in users
  $searcheverything = false;
  if( ( $wgSearchEverythingOnlyLoggedIn && $user-
>isLoggedIn() ) ||
!$wgSearchEverythingOnlyLoggedIn )
    $searcheverything = $user ->
getOption('searcheverything');
    // searcheverything overrides other options
    if( $searcheverything )
    return
array_keys(SearchEngine::searchableNamespaces());
    $arr = Preferences::loadOldSearchNs( $user );
$searchableNamespaces =
SearchEngine::searchableNamespaces();
    $arr = array_intersect( $arr,
array_keys($searchableNamespaces) ); // Filter
    return $arr;
}
```

APPLICATIONS AND SCOPE

There are two major classes of applications that consume, and by extension, microdata:

- Web browsers
- Search engines

Browsers can provide enhanced features by detecting the annotated elements. For instance it can provide to add an event marked up as Event data-vocabulary directly to the user's Google calendar or export it to ICS format.

The other major consumer of is search engines. Instead of simply displaying the page title and an excerpt of text, the search engine could integrate some of that structured information and display it. Full name, job title, employer, address, may be even a little thumbnail of a profile photo. It would definitely catch the attention of everyone.

Google supports microdata as part of their Rich Snippets program [10]. When Google's web crawler parses your page and finds microdata properties that conform to the http://data-vocabulary.org/Person vocabulary, it parses out those properties and stores them alongside the rest of the page data. Google even provides a handy tool to see how Google – sees your microdataproperties.

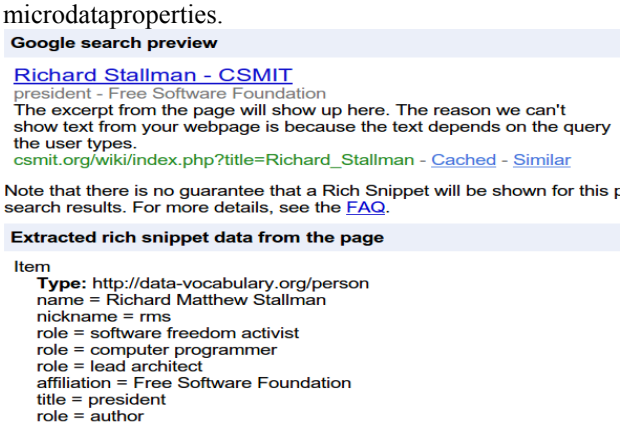

Fig. 8 Screen-shot of Output from Google Rich Snippets tool

And how does Google use all of this information? That depends. There are no hard and fast rules about how microdata properties should be displayed, which ones should be displayed, or whether they should be displayed at all. If someone searches for —Mark Pilgrim, and

Google determines that this – about page should rank in the results, and Google decides that the microdata properties it originally found on that page are worth displaying, then the search result listing might look something like the one shown in the screen-shot below.

The output shown above can be tested at http://www.google.com/webmasters/tools/richsnippets by entering the URL http://csmit.org/wiki/index.php?title=Richard_Stallman in the input field.

CONCLUSION AND FUTURE WORK

The project enhances Mediawiki to recognize the new Semantic Wiki markup developed and to produce microdata annotations accordingly. Thus the huge collection of Wiki pages can be made to serve as a pool of various information, for not only human beings, but also machines.

This can be further extended by making the entire output to be in HTML5, making use of the semantic elements. The search module of Mediawiki is to be enhanced to take advantage of the semantic annotations to provide accurate results with more helpful information than just excerpt of text.

REFERENCES

[1] Vignesh Nandha Kumar K R, Pandurangan N, Vijayakumar R and Pabitha P, *Semantic Annotation of Wiki using Wiki markup for HTML5 Microdata,* International Journal of Engineering Science and Technology, Vol. 2, Issue 12, pp. 7866-7873, 2010.

[2] Mohammed Kayed and Chia-Hui Chang, Member, IEEE, ―FiVaTech: *Page-Level Web Data Extraction from Template Pages*, IEEE Transactions on Knowledge and Data Engineering, Vol. 22, No.2, pp. 249-263, 2009.

[3] Amal Zouaq and Roger Nkambou, Member, IEEE, *Evaluating the Generation of Domain Ontologies in the Knowledge Puzzle Project*, IEEE Transactions on Knowledge and Data Engineering, Vol. 21, No.11, pp. 15591572, 2008.

[4] Jinhyun Ahn, Jason J. Jung, Key-Sun Choi, *Interleaving Ontology Mapping for Online Semantic Annotation on Semantic Wiki*, IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008.

[5] Kensaku Kawamoto, Yasuhiko Kitamura, and Yuri Tijerino Kwansei, Gakuin University, *KawaWiki: A Semantic Wiki Based on RDF Templates*, Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 2006 Workshops)(WI-IATW'06 , 2006.

[6] Adam Souzis, *Building a Semantic Wiki*, IEEE Intelligent Systems, Vol. 20, No. 5 September/October 2005.

[7] *Spinning the Semantic Web*, Edited by Dieter Fensel, James A. Hendler, Henry Lieberman and Wolfgang Wahlster, Foreword by Tim Berners-Lee.

[8] Sebastian Schaffert, Salzburg Research Forschungsgesellschaft , François Bry, Ludwig-Maximilian University of Munich , Joachim Baumeister, University of Würzburg , Malte Kiesel, DFKI GmbH , *Semantic Wikis*, IEEE Software, 2008.

[9] Tim Berners-Lee, James Hendler and Ora Lassila, *The Semantic Web*, Scientific American, May 2001.

[10] Mark Pilgrim, Developer advocate at Google, Inc. Apex, NC, http://diveintohtml5.org/extensibility.html, 2010.

[11] Web Hypertext Applications Technology Working Group, http://whatwg.org/specs/web-apps/current-work/multipage, September 2010.

[12] Mediawiki manual http://www.mediawiki.org/wiki/Manual:MediaWiki_architecture, June 2010.

[13] http://www.mediawiki.org/wiki/Manual:Database_layout