Automated annotation of web page contents for rapid creation of Semantic web contents

Tu Minh Phuong*, Pham Hoang Duy**,and Trinh Huu Kien***
* Departement of Computer Science, Faculty of Information Tecnology,
Posts and Telecoms Insttitute of Technology, Vietnam
Tel: 84-913 507 508 Fax:84-34-511 408 E-mail:phuongtm@fpt.com.vn
** Departement of Computer Science, Faculty of Information Tecnology,
Posts and Telecoms Insttitute of Technology, Vietnam
Tel: 84-912 049 041 Fax:84-34-511 408 E-mail:pdzuy@hn.vnn.vn
*** FPT-Softawre, FPT Co., Vietnam
Tel: (84 4) 6334495 Fax: E-mail: trinhhuukien@yahoo.com

Abstract. The Semantic Web is an extension of the current Web in which information is given formal and explicit meaning. The Semantic Web enables computer programs to understand information contents and thus facilitates more efficient discovery, automation, integration and sharing of data. To create Semantic Web contents one needs appropriate tools. In this paper, we describe such a toolkit we have constructed. The most important feature of the toolkit is that it makes use of information extraction techniques for automatically annotating web page contents. Experiments with a real life application show promising results and demonstrate the usefulness of the toolkit.

Keywords: Semanctic web, information extraction, agent systems

1. INTRODUCTION

With billions of web pages distributed over many countries, World Wide Web (WWW) provides ideal means to present and to access digital information. However, this huge volume of web documents cause difficulties in retrieving and sharing information over WWW. At present, most of the information in WWW is presented in natural language form (using HTML). This presentation method is suitable for human understanding but is difficult for automated processing. Computer applications can not understand data and information presented as natural texts.

In order to solve this problem, research organizations and commercial companies have co-cooperated in the development of Semantic Web. According to the director of World Wide Web Consortium (http://www.w3c.org) Tim Berners-Lee, the father of WWW, Semantic Web is an extension of the current WWW, in which information is given formal and explicit meaning so that computer programs can understand and process these information more effectively [1]. As a result, the Semantic Web is composed of traditional web pages and explicit semantic descriptions. Adding semantic information provides further knowledge to computer applications (software agents) in order to improve the quality of classifications, retrievals and exchange of information.

To create Semantic Web contents one needs appropriate tools. In this paper, we describe such a toolkit we have constructed. The most important feature of the toolkit is that it makes use of information extraction techniques for automatically annotating web page contents. In order to demonstrate and evaluate the toolkit, we describe an application with search features based on meaning of information contents among web pages generated by the toolkit.

2. COMPONENTS OF SEMANTIC WEB

In order to describe functionalities of our toolkit, in this section we give a brief overview of the components of the Semantic Web. The major components of the Semantic Web are:

Ontology and languages, which present semantics of information.

- Tools for creating the infrastructure of the Semantic Web.
- Applications of semantic web.

2.1. Languages for Semantic Web

The well-known and broadly used mechanism, which enables sharing and exchange common understandings of information, is *ontology*. Ontology is an explicit descriptions of concepts within a given application domain and relationships between these concepts. Ontology provides a common vocabulary for information exchange between applications and Web services. Specific instances of the concepts defined in the ontologies – instance data - together with ontologies constitute the basis of Semantic web. In recent experiments to prototype the Semantic Web, researchers from different communities have proposed a number of languages for representing ontologies and instance data. The most well known among those languages are RDF and RDFS [2], DAML+OIL [8,9].

RDF and RDF Schema. RDF (Resource Description Framework) mechanism allows describing meta-data. RDF considers objects on the Web such as web pages,

paragraphs as resources. Each resource is described by a triple *object-attribute-value*. For example, the statement "Phuong is the author of a given web page" could be presented by a tuple: (http://www...., author, "Phuong"). RDF Schema (RDFS) is a simple type of system for RDF. It provides a mechanism to define domain specific properties and classes of resources to which those properties can be applied. . The object definition including attributes and relationships is necessary to construct ontology.

DAML + OIL. RDF and RDFS only allows to present semantics at a simple level. In case of numerous objects having complex logical relationships with each other, it is required to use more powerful presentation tool. DAML (Darpa Agent Markup Language) and OIL (Ontology Interface Layer) are built to meet such requirements. DAML+OIL is an extension of RDFS. In DAML+OIL, semantics is illustrated by descriptive logic, which enables to use BOOL logic for describing relationships between objects and offering more basic relationships than RDFS.

2.2. Tools for the Semantic Web

To create and use the Semantic Web, it is required to have the following support tools.

Tools for creating and intergrating ontology. These tools allow generating concepts, attributes of these concepts, relationships and hierarchy among these concepts. They are often equipped with graphic user interface and comply standards of web applications. A typical example is Protégé [11].

Annotation tools. These tools support the creation of semantics, which means concrete values of concepts, attributes and relationships, from normal data according to a given ontology. Generated values could be presented by several languages mentioned above. At present, most of these tools only support manually creating annotations, therefore, this process is time-consuming [6].

Repository. Once successfully generated, ontology and semantics components should be stored in repositories. These repositories essentially are databases, which hold descriptions written by languages like RDFS or DAML+OIL and allow transforming queries of such languages into SQL ones. Sesame [7] is a very typical among such repositories.

Reasoning service. This service supports to retrieve concrete values of concepts or attributes corresponding to the ontology in repositories. An example system is Ontobroker [5], which is now commercialized.

2.3. Applications

The Semantic Web enhances the functionalities, intelligence, and automation of several existing applications. The domains of applications, which are the promise land for the Semantic Web, comprise web services, knowledge managements, and electronic commerce [3].

Web services can be applications and devices, which provide access via WWW infrastructure. The Semantic Web provides necessary information and knowledge for searching, interacting, sharing and integrating web services.

Knowledge management involves in collecting, storing, retrieving, accessing and providing information and knowledge within organizations in order to fully exploit knowledge assets of those organizations. Such tasks require numerous functionalities, which are more comprehensive than document management systems or databases, like intelligent retrieval, automated information extraction from texts, database associations, and automated document summarization. Those functionalities can be employed in the infrastructure supported by semantic web.

The amazing growth of *e-commerce* leads to the enormous volume of transactions on network. To automate those transactions, software applications are required to support: transforming documents across platforms in electronic transactions; ontology describing merchandises and services which enable agent to search, classify and negotiate.

3. OVERVIEW OF THE TOOLKIT

The major goal of our toolkit is to support the process of generating, storing, and querying semantics of web pages. This process requires supports of numerous specific tools. Although some of such tools already exist, we believe that integrating those tools into a single system (with certain adaptations) is essential to comprehensively support generating and querying the Semantic Web.

Besides existing tools, the system includes several components created by the authors. The most important one is the module for automated annotation using information extraction techniques. The details of the modules are given in the next section. To demonstrate the usefulness of the toolkit, we have added modules which perform search based on generated semantics into the system. The components of the whole system are depicted in Figure 1.

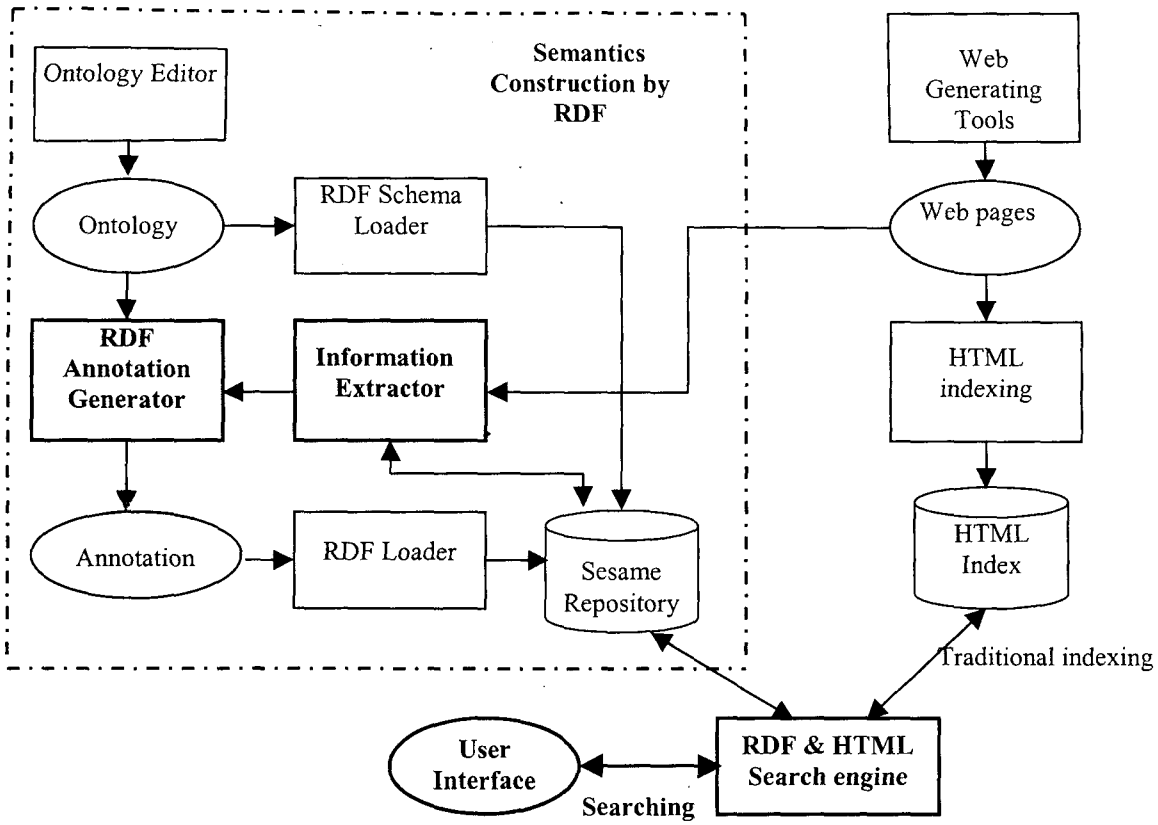

Figure 1. System for generating semantic web and applications

In the above figure, the rectangles present function components, the ellipses present information or data generated by these components. The bold rectangles with dark backgrounds are built by the authors; the other rectangles depict existing components integrated into our system. Those components include ontology editor Protégé [11], repository of RDF descriptions Sesame [7], RDF and RDFS loader, traditional search engine based-on keywords.

The semantics construction is performed by the modules in the dashed rectangle at the upper-left corner. This is major part of the system. Generated semantics is employed in intelligent search applications represented at the lower of Figure 1. To comply with searching on traditional web pages (without semantics), the system includes the module of HTML indexing by keywords (at the upper-right of Figure 1).

The system functions as follows.

- Firstly, users construct an ontology for a given application domain by ontology editing tools. Then, the ontology is transformed into RDFS descriptions and is stored in the Sesame repository.
- Once successfully creating the ontology, the next step is to annotate web pages, which means adding the semantics to web pages by filling values from web pages to concepts and attributes in the ontology. Typically, the annotation process is done manually. In case of enormous volume of web pages, this process is time-consuming and error-prone such as missing or un-accurate annotations. Our toolkit is aimed to solve this problem by using information extraction from web pages and automating annotations. To annotate a web page, the web page is passed to the extraction module. Based-on the ontology, this module extracts from the page information about concrete values of concepts and attributes described in the ontology.
- The information extracted from the above steps goes to the module of annotation generating. This module has responsible to create a RDF tuple describing extracted information and to pass these descriptions to Sesame repository.
- In parallel with above steps, the web page is indexed by keywords as the traditional fashion.
- Finally, semantics is employed in the search engine. This engine uses RQL to query the repository, also coordinates with the deductive mechanism based-on semantics to provide intelligent results of searching. The query can be in the form of natural languages. In such case, the semantics of the query is extracted by the technique employed in the component, which extracts information for the annotation process.

4. EXTRACT INFORMATION FROM TEXTS AND AUTOMATED ANNOTATIONS

The goal of the information extraction module is to discover information, data corresponding to concepts in the ontology, extract that information, and pass them to the annotation generation module. For example, examine the following text from a web page of job announcement.

A software company needs programmers for a E-commerce project. Prospective candidates should have at least four years of programming experience in Windows and Unix. Skills of programming languages Java and Javascript are required, especially the candidates should have three or more years programming experience in Java. Special considerations will be given to the candidates familiar with Oracle.

Suppose that ontology stores concepts, attributes, and relationships described in Figure 2. The extraction process will give the following result:

```
Profession: programmer
programmer: Experience :four year
Skills:
        Operating system: Windows,
Unix
        Languages: Javascript
                   Java: Experience
:three year
```

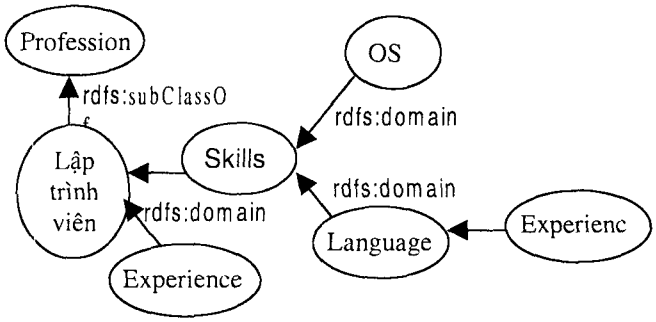

Figure 2: A simple sample of ontology

There is a number of extraction techniques introduced in [4,10,12]. In our case the documents to be annotated often are weak-structured (written in natural languages), and extracted information should have structures specified by the ontology. Therefore, we choose the technique described in [4]- this technique can best satisfy both of the requirements. We have also made additional modifications to the original algorithm to meet the specificities of our problem [13].

Extraction process comprises the following steps:

Step 1: Identify constants and keywords. Constants are concrete values of concepts or attributes stored in the ontology. Keywords are terms or phrases, which allow specifying constants possibly belonging to given concepts or attributes. For example, in the above sample, "Java" is a constant and "programming language" is the keyword indicating this constant belongs to the language attribute of the concept of "skills"

Constants and keywords are identified by using rules. The rules are patterns presented in the forms of regular expressions (like in Perl), but are enhanced by using vocabularies. For instant, the pattern identifying experience is as follows:

```
Programmer:      Experience      case
insensitive
        constant {extract Number, "[a-
zA-Z\s]*\s+year" };
        lexicon      {Number      case
insensitive, filename "number.dat" };
        keyword {"\bexperience\b" }
end;
```

The pattern indicates the "Experience" attribute of "programmer" can be recognized by the expression starting with one "Number", ending with "year"; "Number" is a term stored in file named "number.dat" (this term lists strings of "one", "two", "three" etc.); The following term is "\bexpericence\b".

The patterns identifying constants and terms are stored in the ontology with descriptions of concepts and attributes. Hence, the ontology is expanded in order to contain this additional information.

At the beginning of the extraction process, all of the patterns in turn are employed to retrieve constants and terms in the documents. The results of identified constants and terms are stored in the table described in step 2.

Step 2: Create table Name|Value|Position. The constants and terms, identified in the above step, are stored in a table. Each row of the table contains the name of the concept or attribute corresponding to constant or term, value (retrieved from the documents), starting and ending positions in the documents. Terms are distinguished from constants by the prefix of keyword at the beginning. For example, from the document in the above sample, we can construct the following table (just a snapshot)

```
...
programmer:Experience|four
year|108|117
language:experience|                four
year|108|117
KEYWORD
programmer:language|experience|134|14
4
Skills:OS|Windows|148|155
Skills:OS|Unix|161|165
KEYWORD     Skill:language|programming
language|175|196
Skills:language|Java|198|202
Skills:language|Javascript|206|216
Skills:language|Java|316|320
...
```

Step 3: Create information corresponding to the ontology from the table. At this step, the information from the table Name|Value|Position are employed to create values for concepts and attributes in the table. Essentially, this step aims to solve contradictions or ambiguousness of the information in the table by applying several heuristics rules. For example, in the table, we find that "four year" recognized at step 1 could belong to the experience of general programming but also to a specific programming language because this fact can match with both patterns of these two attributes. Likewise, "Java" can be recognized twice, but only one value can store in repository. For those situations, we apply following heuristics:

- If a concept or attribute only allows having one value but in the table, there are many values then the value, which is closest to the corresponding term. Instantly, in the table there are two constants for attribute "programmer: experience" including

"three year" and "four year". The first heuristics eliminate the value "three year" because of its longer distance to the term "programmer : experience"

- If constants are similar then only the constant, whose distance to the corresponding term is shortest, is chosen. For example, the table has two constants "four year" then the constant corresponding to the term of "programmer : experience" is stored because of its shorter distance to the term.
- If there are many overlapped pairs of constants/terms then the longeest pair is chosen. For instant, the term "programming experience" is overlapped with the term "experience" but the former is longer and thus "programming experience" is chosen.
- If the concept can only have one value then the first constant appearing in the table, is retained.
- The relationships of one-to-many are often represented by overlapped constants in the documents.

In the above rules, distances, used to compare, are calculated by positions, where constants and terms appear in the documents. After applying heuristics, remaining constants are passed to the annotation generation in order to transform into RDF forms.

5. APPLICATION DEPLOYMENT AND EXPERIENCE

5.1. Application deployment

The system is deployed as a Web application, which enables to build and to store semantics centrally in a server or a workstation. The deployment process uses following tools and programming languages.

Jave is used because of its advantages: object-oriented language; platform independence; easy to integrate to web applications by supporting Servlet/JSP; easy to connect to databases via JDBC. Moreover, Java (from version 1.4) offers built-in regular expression tools, which is very useful and essential for extraction techniques.

This system comprises two databases: one is for Sesame repository; the other- administrative information of the system. Both are built by MySQL, which is an open-source database and runs rather fast, does not require many system resources as well.

Web server runs Tomcat 4.1 (http://jakarta.apache.org/tomcat), a free web server, which supports Servlet/JSP.

Indexing web pages and keyword searching are implemented by using Jakarta Lucene search engine (http://jakarta.apache.org/lucene). This is an open source package written by Java and offers many search features based on keywords.

5.2. Experience

With the aim of demonstration and experience, the system is employed to generate annotations of web pages, which present programmers' personal information and their skills, and then the search component provides searching their information based-on keywords and semantics. First, the ontology of the programming profession, related skills and experience is constructed. The ontology is only built once for all of the web pages.

After having the ontology, users enter the address of the web pages required to annotate via the system interface. At this step, users can create a brand new web page and its annotations, or generate annotations for the existing web page by loading it to the system. The system interface used to enter the web page address for producing annotations is in Figure 3.

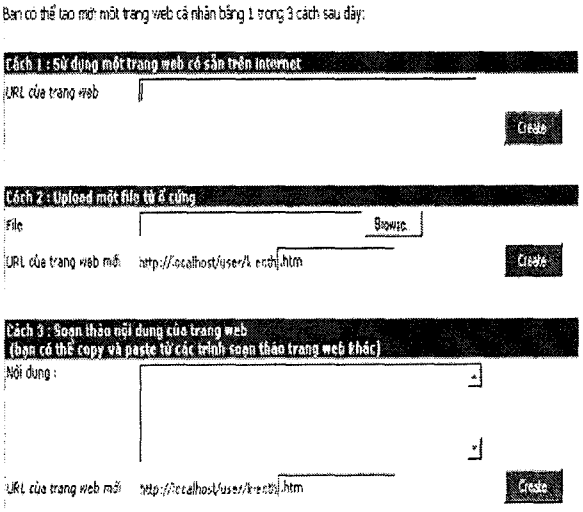

Figure 3. Form for entering address of the web page which require annotations

After confirming the web page for annotating and clicking on "Create" button, the extraction component generates annotations. Users can view the annotations and modify them, as they need. Figure 4 illustrates the annotations of programmers' skills generated for a sample web page.

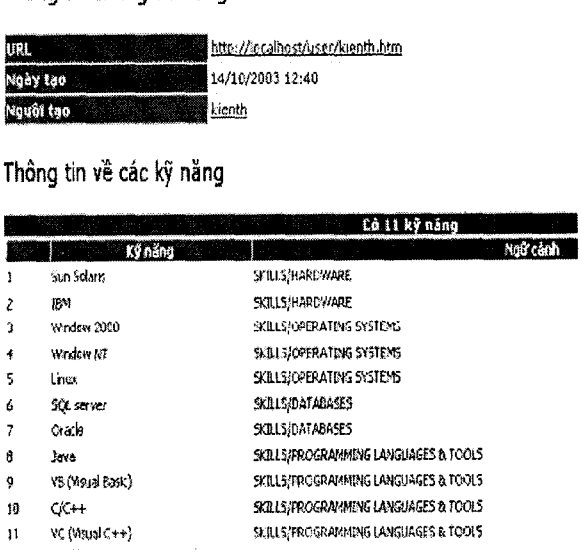

Figure 4: Annotations of skills extracted from web pages

After completing annotations, users can perform information searching by keywords and/or semantics as shown in Figure 5.

To experience the system, we use 30 personal home pages of programmers, who are currently working at FPT-Software, FPT Co. (http://www.fpt.com.vn) and several pages from Internet. These pages are automatically annotated by our system. The results are then compared with the annotations produced manually. The results are evaluated by two criteria: recall (number of extracted data terms/number of data terms in documents), precision (number of correctly extracted data terms/number of extracted data terms). Empirical results show that the quality of constructed ontology has strong influence on those of annotations. After tuning the ontology, with 30 personal web pages, the values of recall and precision are 88% and 95% respectively. Such recall and precision are rather high, and they match to features of chosen technique of information extraction. The results of automated annotations can be tuned in order to archive the best performance.

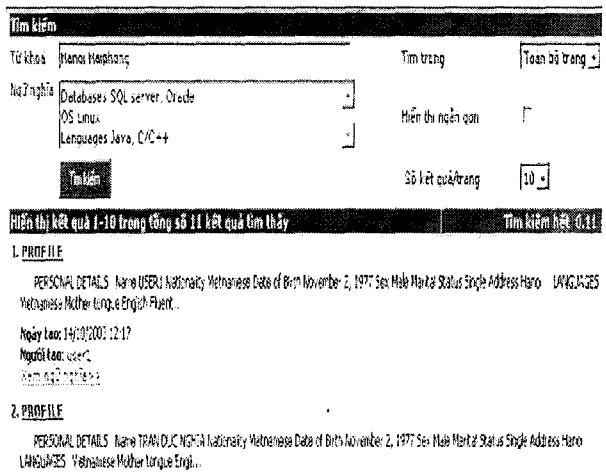

Figure 5: Search results, which are combined between keywords and semantic techniques

6. CONCLUSIONS

This paper presents the design and the implementation of a toolkit for creating Semantic Web pages. Experiments with the toolkit shows that information extraction techniques significantly reduce time for creating annotations of web pages. In facts, this is the most time-consuming step in creating Semantic Web contents. Empirical results prove a rather high precision of the automated annotation process. Also, the adaptation and the integration of existing tools to our toolkit not only accelerate the implementation process, but also provide more features and convenience to users compared to single tools. However, our tools do not offer several automated features such as automatically generating ontology from texts. Therefore, the further developments are focused on integration and improving these features.

7. ACKNOWLEDGEMENT

The research was supported by The Natural Science Council of Vietnam.

8. REFERENCES

[1] T. BERNERS-LEE, J. HENDLER, O. LASSILA, The Semantic Web, Scientific American, May 2001.

[2] D. BRICKLEY, R.V. GUHA, Resource Descr ption Framework (RDF) Schema Specification, World Wide Web Consortium, Proposed recommendation 2001.

[3] Y. DING, D. FENSEL, M. KLEIN, B. OMELAYENKO, The semantic web: yet another hip? Data & Knowledge Engineering 41, Elsevier 2002, pp 205–227.

[4] D.W. EMBLEY, D.M. CAMPBELL, R.D. SMITH, S.W. LIDDLE, Ontology-Based Extraction and Structuri ng of Information from Data-Rich Unstructured Documents, Proc. of 1998 ACM Inter. Conf. on Inform. and Knowledge Man., CIKM 1998, USA, pp 52-59

[5] D. FENSEL, S. DECKER, M. ERDMANN, H.-P. SCHNURR, R. STUDER, A. WITT, Lessons le.rned from applying AI to the web, Journal of Cooperative Information Systems 9 (4) (2000).

[6] S. HANDSCHUH, S. STAAB, CREAM – Cre ating metadata for the semantic web, Computer networks. vol. 42, Elsevier 2003, pp 557-571.

[7] http://sesame.aidministrator.nl/.

[8] http://www.ontoknowledge.org/oil.

[9] [9] http://www.daml.org.

[10] N. KUSHMERIC, Wrapper induction: efficiency and expressiveness, Artificial intelligence, vol.118,2000.

[11] N. F. NOY, M. SINTEK, S. DECKER, M. CRUBEZY, R. W. FERGERSON, M. A. Musen, Creating sem.ntic web contents with Protégé -2000, IEEE Intelligent systems, 3-4/2001, pp 60-71.

[12] S. SODERLAND, Learning information extraction rules for semi-structured and free text. Machine learning, 34. Kluwer Academic Publishers.(1999)

[13] TU MINH PHUONG, Information Extraction and Evaluation of Candidates with Fuzzy Set techniques, Proc. of Inter. Conf. on Fuzzy system. and Knowl. discovery, FSKD 2002, Singapore, 2002, pp 481-485