An Ontology-Supported Web Focused-Crawler for Java Programs

Sheng-Yuan Yang[1] Chun-Liang Hsu[2]
[1]*Dept. of Computer and Communication Engineering, St. John's University, Taiwan, R.O.C.*
[2]*Dept. of Electrical Engineering, St. John's University, Taiwan, R.O.C.*
E-mail: {[1]ysy, [2]liang}@mail.sju.edu.tw

Abstract

This paper proposed an ontology-support web focused-crawler: OntoCrawler III for Java programs, in which only the user entered some keywords would the system supported by the domain ontology actively provide comparison and verification for those keywords so as to up-rise the precision and recall rates of webpage searching. This technique has practically been installed in Google and Yahoo search engines and furthermore searched and filtered out unduplicated and related Java open source webpages and accordingly downloaded and stored the results into a database to let the backend systems to do advanced processes. The preliminary experiment outcomes proved the OntoCrawler III based on ontology-supported techniques proposed in this paper could not only really up-rise the precision and recall rates of webpage searching but also should successfully download related webpage information.

1. Introduction

In the information explosion era of the Internet, how to find out the advantage Java Programs from the vast rolling torrent of the Internet that just likes to search a needle from the haystack and that work is usually in a maze to users. However, how to search advantage information has become essential topic for discussion in users or even the search engine manufacturers themselves [12]. Therefore, the active webpage searching technique plays an important basic component in the contemporary information system.

October in 1998, Google was officially born and soon became the most satisfying searching engine contemporarily for the users. That Google adapted query method with keywords which resulted in a hard-breathing searching outcome even with less keyword. In such long and complicatedly listed outcomes, it caused users not only took more time to look up those information but also indicated the query system itself couldn't comprehensively know the true query intention of users. The main cause was that the keywords entered by users were not completed and not able to obviously indicate the query demands of users. Furthermore, there are so many keywords being the same words with different meaning in different fields. The system would finally produce many complicated cross-field query outcomes when system didn't respectively managed to classifying query requisition and specifying fields [11]. This condition always let users not able to select real information they need. Hence, how to make the webpage crawling techniques more precise and clarify executing search what information user really need has become an essential issue between users and information systems.

Along with popularity of application and use of Internet, people who want to search information they need on Internet have to run on different independent query engines and enter keywords to accomplish getting needed information. To make users with faster and more effective way get advantage information and knowledge from huge amount of Web information. Hence, we wanted to design an integrated Focused-Crawler which can assist and release the loading of query works of users as well as support the core of webpage search systems so as to improve the system performance.

To sum up, the main purpose of this paper was to employ the ontology technique to design the ontology of Java programming codes and construct the analyze ontology classes of these codes through the ontology construction tool Protégé [4], and then accompanied with MS SQL Server database to set up an ontology sharing platform of some keywords of Java programming codes. Finally, we use Java [7] to build up the OntoCrawler III (Ontology-supported Focused-Crawler). In other words, Introducing keywords comparison and judgment of Java programming codes ontology not only exclude the extra webpages resulting from the same word with different meaning but also up-rise the precise rate and recall rate of query webpage. In addition, those searched Java programs and their related webpage information were stored by the crawler for saving lots of searching operation time of users as well as providing support for both query works of users and core of webpage search systems.

2. Background Knowledge and Techniques

2.1. Ontology

Ontology was one theory in philosophy and primarily to explore knowledge characteristics of life and real objects. It can provide complete semantic models with sharing and reusing characteristics. To describe the structure of the knowledge content through ontology can accomplish the knowledge core in a specified domain and automatically learn related information, communication, accessing and even induce new knowledge; hence, ontology is a powerful tool to construct and maintain an information system [14]. Figure 1 illustrates the ontology structure of Java programming codes, which defines related basic knowledge of Java and its conceptual hierarchy relationship and relevant features.

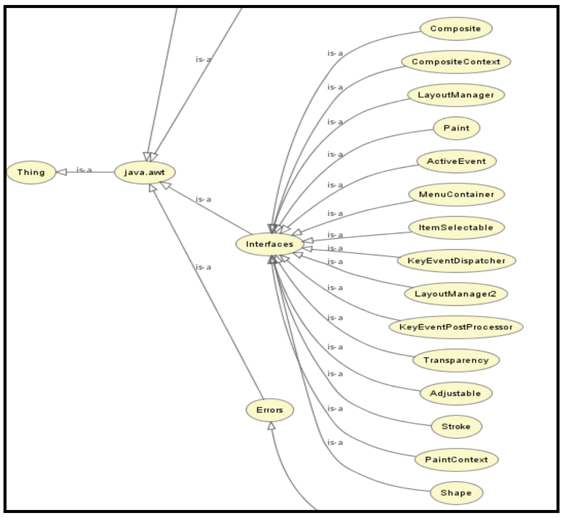
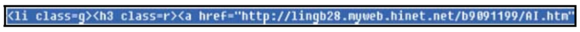


Figure 1. Part of ontology structure of Java programming codes

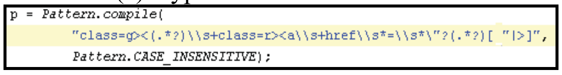
2.2. Regular Expression

Regular expression is a character queue to describe specified order. The descriptive style, so to call pattern, could be used to search matched pattern in another character queue. Regular expression can use universal words, set of words, and some quantifiers as specifying ways [10]. There were two supported classes for this expression: Pattern and Matcher, and we would use Pattern to define a Regular expression. If we want to conduct pattern matching with other character queue, we would use Matcher. Figure 2 showed an example program adapting regular expression in this system.



```
<li class=g><h3 class=r><a href="http://lingb28.myweb.hinet.net/b9091199/AI.htm"
```

(a) Hyperlink format in HTML



```
p = Pattern.compile(
        "class=g><(.*?)\\s+class=r><a\\s+href\\s*=\\s*\"?(.*?)[ "|>]",
        Pattern.CASE_INSENSITIVE);
```

(b) Corresponding regular expression

Figure 2. Example on regular expression

2.3. Developing Techniques

The developing tool of this system is Borland JBuilder. It is an integrated development environment of Java, which have a fine human-machine interface and code debugging mechanism to make a fast integration of each code block when the system was developed, and accordingly reduce the time of system development. In addition, Java [7] provides lots of functions and methods to integrate web applications and databases. In the view of extensibility, Java is absolutely the optimal choice for solving the problem of cross platform.

This system adapted MS SQL Server as backend knowledge-database sharing platform based on ontology. MS SQL Server is one mostly used relational database management system. SQL (Structured Query Language) is one query language to get the data in the database. The ontology construction tool, Protégé, was an ontology freeware developed by SMI (short for Stanford Medical Informatics). Protégé not only was one of the most important platforms to construct ontology but also the most frequently adapted one [5,6]. Protégé was adapted in this paper and its most special feature is that used multi components to edit and make ontology and led knowledge workers to constructing knowledge management system based on ontology; furthermore, users could transfer to different formats of ontology such as RDF(S), OWL, XML or directly inherit into database just like MySQL and MS SQL Server, which have better supported function than other tool [15].

3. System Architecture

3.1. Construction of Ontology Database

Nowadays the research on ontology can be branched into two fields: one is to configure huge ontology in a specified field and through them to assistant the knowledge analysis in this field; the other is to study how to construct and express precisely with ontology. In this paper, we adopted the former in which took advantage of built ontology database of some Java programming codes to support whole system operation. In the mentioned ontology database, it included two constructing stages [13,15]; one is statistics and analysis of related concepts of Java programming codes, the other is construction of ontology database. First of all, we conducted statistics and survey of related Java programming codes to fetch out the related concepts and their synonym appearing in those programs and employed the ontology construction tool Protégé to construct that ontology. The second stage of ontology construction is the construction of ontology database of Java programming codes, in which the main part work is to transfer the ontology built with Protégé into MS SQL database for conveniently processing by the system.

3.2. System Structure of OntoCrawler III

Figure 3 illustrated the operation system structure of

OntoCrawler III, and related techniques and functions of every part were detailed below.

(1) Keyword & Download Directory: contains the preprocess of executing query webpage, including to empty the output area, transfer input characters into URI code and then embedded into Google's/Yahoo's query URL, transfer the input string of the default downloading location into the file name of the storage location and empty that field, and finally the system would remind users to input related default operations [14].

(2) LinkToGoogle & Yahoo: declares an URL and add Google/Yahoo query URL on well transferred URI code, and then used a BufferedReader to read and used while loop to add String variable "line" line by line. Finally, output "line" as text file as final analysis reference. The file content was the html source file of the webpage.

(3) RetrieveLinks: uses regular expression to search for whether there are matched URL from the variable "line." The matched ones can be downloaded outputted the txt file to provide the system for further processing.

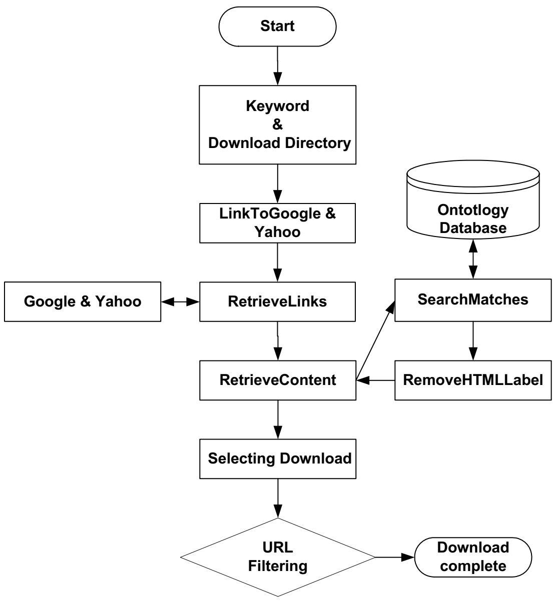


Figure 3. System architecture of OntoCrawler III

(4) RetrieveContent: owing to wasting too much time on RetrieveContent processing and the executing order would be a problem to make system interface entirely stopped. So, the system designing with "thread" got free from Swing thread events, and this made it possible to do some proper change of the interface when querying webpage. And then we used BufferedReader to read in "RetrieveLink" with "while" loop line by line, that meant we checked one URL link once a time and really linked the URL. After judging what kind coding of the webpage was, we read in the html source file of webpage with correct coding and output it as text file so as to let system conduct further processing. After completing all procedures mentioned above, we could used SearchMatches method to judge whether the webpage was located in the range we hoped to query; supposed the answer was "yes", we would execute RemoveHTMLLabel to delete the html label from source file and remained only the text content so as to let system conduct further processing and analyzing. Finally, we collected the number of queried webpage and divided with total of the webpage and the mean we got was the percentage of query processing, detailed procedure as shown in Figure 4.

```
CrawlerThread=new Thread   (new Runnable(){
public void run(){
Crawling=true;
ProgressBar.setValue(0);
EnterButton.setText("停止");
InputTextField.setEnabled(false);
StringReader sr=new StringReader(RetrieveLinkStr);
BrowsedValue=1;
PageURLStr="";
try{
BufferedReader br=new BufferedReader(sr);
while((PageURLStr=br.readLine())!=null&&Crawling){
PageURLStr=PageURLStr.replace("null","");
String s2;
try{
URL u=new URL(PageURLStr);
URLConnection URLConn=(HttpURLConnection)u.openConnection();
URLConn.setRequestProperty("User-agent","IE/6.0");
BufferedReader br1;
br1=new BufferedReader(new InputStreamReader(URLConn.getInputStream()));
s2="";
CrawlingPageContentStr="";
boolean UTF8=false;
while((s2=br1.readLine())!=null){
Pattern p1=Pattern.compile("(?s)charset\\s*=\\s*utf-8",Pattern.CASE_INSENSITIVE);
Matcher m1=p1.matcher(s2);
Pattern p2=Pattern.compile("(?s)charset\\s*=",Pattern.CASE_INSENSITIVE);
Matcher m2=p2.matcher(s2);
if(m1.find()){
UTF8=true;
break;
}
else if(m2.find())
break;
}
URLConnection URLConn1=(HttpURLConnection)u.openConnection();
URLConn1.setRequestProperty("User-agent","IE/6.0");
if(UTF8)
br1=new BufferedReader(new
InputStreamReader(URLConn1.getInputStream(),"utf-8"));
else
br1=new BufferedReader(new InputStreamReader(URLConn1.getInputStream()));
while((s2=br1.readLine())!=null)
CrawlingPageContentStr=CrawlingPageContentStr+s2+"\r\n";
}
catch(Exception FailURL){
CrawlingPageContentStr="invalid connections or connecting overtime !!";
}
FileWriter gg=new FileWriter("Content"+BrowsedValue+".txt");
gg.write(CrawlingPageContentStr);
gg.close();
RemoveHTMLLable();
SearchMatches();
if(Matches){
ResultLabel=new JLabel("Matched Gathering Conditions");
ResultLabel=new JLabel(MatchStr);
ResultLabel=new JLabel("Matched Keywords");
setButton(PageURLStr);
ResultPanel.validate();
ResultScrollPane.validate();
FileWriter fw=new FileWriter("Matched Content"+BrowsedValue+".txt");
fw.write(MatchPageContentStr);
fw.close();
}
ProgressBarValue=(int)100*BrowsedValue/TotalPageValue;
ProgressBar.setValue(ProgressBarValue);
BrowsedValue++;
}
br.close();
}
catch(IOException RetrieveContent){}
RetrieveLinkStr="";
ResultLabel=new JLabel("Webpage Retrieved Completely !! ");
InputTextField.setEnabled(true);
EnterButton.setText("Crawling");
Crawling=false;
```

Figure 4. Procedure of RetrieveContent

(5) Selecting Download：clicks the button "前往此頁面!", means "go to this webpage !", when the system finished 100% processing and then

displayed the corresponding webpage content generated by the function RetrieveContent().

(6) URL Filtering: employs regular expressions to find out whether the matched URLs exist from the variable "line." If yes, the system carried out the downloading action and stored the returned result into the default location chosen by users.

4. System Verification and Comparison

Given an example with the keyword: cell-phone programs, we explain how to search the related webpages of Java open source code and detailed follows. The equations (1) and (2) to define Precision Rate, R_P and Recall Rate, R_R, respectively [13], in which NW_T meant the number of total returned webpages; NW_C meant number of correct returned webpages; NW_R meant number of related returned webpages but they were not necessarily the correct webpage. After comparing returned webpage by domain experts one after another, Table 1 illustrates the average R_P and R_R of Google were 6% and 67% while Yahoo were 6% and 60%, respectively.

$$R_P = \frac{NW_C}{NW_T} \tag{1}$$

$$R_R = \frac{NW_C}{NW_R + NW_C} \tag{2}$$

Table 1. Comparison of the front 100 queries on Google and Yahoo

	NW_C	NW_R	NW_T	R_P	R_R
Google	6	3	100	6%	67%
Yahoo	6	4	100	6%	60%

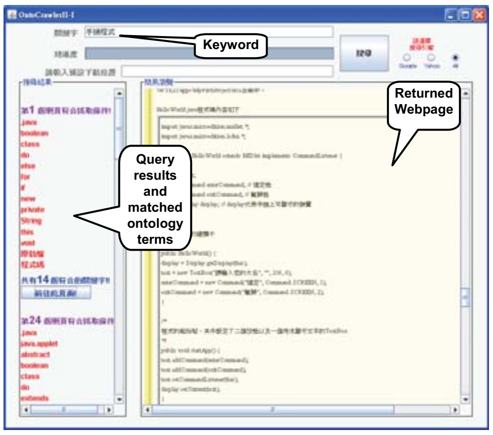


Figure 5. Returned screen of OntoCrawler III

Table 2. Search results of OntoCrawler III

Sampling = 9	NW_C	NW_R	NW_T	R_P	R_R
OntoCrawler III	8	0	8	100%	100%

When we used Google and Yahoo as base of webpage query, but on OntoCrawler III we keyed in the same keyword. The returned screen was shown in Figure 5 and the comparison results were shown in Table 2. After comparing Tables 1 and 2, the query precision and recall rate for Google researching engine through assistance of OntoCrawler has up-rise around 94% and 33% ($\frac{100\% - 6\%}{100\%}$, $\frac{100\% - 67\%}{100\%}$) while conditions of Yahoo were about 94% and 40% ($\frac{100\% - 6\%}{100\%}$, $\frac{100\% - 60\%}{100\%}$). From the above comparison, it indicated that OntoCrawler III offered more precision and recall rate than Google and Yahoo on webpage searching for Java open source codes; in addition, the technique we proposed has its availability.

5. Related Works and Comparisons

Topical crawling was first introduced by Menczer [8], which can automatically traverse Internet and retrieve webpages by hyperlinks. A focused crawler or topical crawler is a Web crawler that attempts to download only webpages that are relevant to a pre-defined topic or set of topics, which was first introduced by Chakrabarti et al [1]. In the face of the inundant spam websites, traditional web crawlers cannot function well to solve this problem [3]. Ontology is a technology for conceptualizing specific domain knowledge, which can provide machine-readable definitions to the domain. Therefore, ontology should be utilized to enhance the performance of focused crawlers by precisely defining the crawling boundary. Here are many examples of such crawler systems. Dong et al [2] exhibited a conceptual framework of an ontology-based focused crawler serving in the domain of transport services. Xing [9] proposed a framework and algorithm of the ontology-based adaptive topical crawling which used the ontology technology to reduce the crawler to get the unrelated information for improving the correlativity of the topical crawler. Focused crawler technologies in general and ontology-based approaches in particular are considered the foundation for the next generation of information services. In this paper, we proposed the use of ontology-supported technique to provide a semantic level solution for a focused crawler named OntoCrawler III so that it can provide fast, precise and stable query results. The technique in this research has practically applied on Google and Yahoo searching engines to actively search for webpages of related information and the experiment outcomes indicated that this technique could definitely up-rise precision rate and recall rate of webpage query.

6. Conclusion

This paper have proposed an ontology-support web focused-crawler: OntoCrawler III for Java programs, in which only the user entered some keywords would the system supported by the domain ontology actively provide comparison and verification for those keywords so as to up-rise the precision and recall rates of webpage searching. This technique has practically been installed in Google and Yahoo search engines and furthermore searched and filtered out unduplicated and related Java open source webpages and accordingly downloaded and stored the results into a database to let the backend systems to do advanced processes. The preliminary experiment outcomes proved the OntoCrawler III based on ontology-supported techniques proposed in this paper could not only really up-rise the precision and recall rates of webpage searching but also should successfully download related webpage information.

The OntoCrawler III not only inherited the related functions of the OntoCrawler I (applying domain on Scholars ontology) [15] and the OntoCrawler II (applying domain on call for papers ontology) [13], developed by the intelligent systems laboratory, department of computer and communication engineering, St. John's university, Taiwan, but also extended its applying domain on Java programming codes and enhanced and improved related system execution performance, detailed as shown in Table 3. This also proves the OntoCrawler architecture can easily suit to the corresponding applying domain through changing the system ontology so as to test and verify the OntoCrawler architecture we proposed has its domain robustness.

Table 3. Performance comparisons among different versions of OntoCrawler

	OntoCrawler I	OntoCrawler II	OntoCrawler III
Search Engines	Only Google or Only Yahoo	Google & Yahoo	Google & Yahoo
Filtering Technique	Individually show webpages and non-filtering	Filtering duplication webpages and integration display	Filtering duplication and no any programming codes webpages and integration display
Execution Performance	Around 60 minutes	Around 40 minutes	Around 30 minutes
Download Function	No	No	Yes
Database Reading	Can partially read	Can partially read	Can completely read

This OntoCrawler III could easily combine with backend systems due to both an open-sourced code design philosophy and more developing tools adapted by enterprises so as to attract related researchers to use this technique. As to the extension capability of the system, users could transfer it into other domain OntoCrawler III only by replacing its ontology, and the query subjects could combine with other searching engines by clarifying the meaning of related query sentences/words and then a query base could easily achieve. Continuously improving the performance efficiency, expanding database of ontology and its related linking interface, and developing the middle programs with backend systems would be the everlasting research in the future.

Acknowledgement

The authors would like to thank L.W. Chen, M.Y. Tsai, and M.S. Tsai for their assistance in system implementation and related experiments. This work was supported by the National Science Council, Taiwan, R.O.C., under Grant NSC-98-2221-E-129-012 and the Ministry of Education, Taiwan, R.O.C., under Grant Skill of Taiwan (1) Word No. 0990045921s.

References

[1] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused Crawling: A New Approach to Topic-specific Web Resource Discovery," *Proc. of the 8th International Conference on World Wide Web*, Toronto, Canada, 1999, pp. 545-562.

[2] H. Dong, F.K. Hussain, and E. Chang, "A Transport Service Ontology-based Focused Crawler," *Proc. of the 4th International Conference on Semantics, Knowledge, and Grid*, Beijing, China, 2008, pp. 49-56.

[3] H. Dong, F.K. Hussain, and E. Chang, "A Survey in Semantic Web Technologies-Inspired Focused Crawlers," *Proc. of the 3rd International Conference on Digital Information Management*, London, United Kingdom, 2008, pp. 934-936.

[4] J.H. Gennari et al., "The evolution of Protégé: and environment for knowledge-based systems development," *International Journal of Human-computer studies*, 58, 2003, pp. 89-123.

[5] W.E. Grosso, H. Eriksson, R.W. Fergerson, J.H. Gennari, S.W. Tu, and M.A. Musen, "Knowledge Modeling at the Millennium: the Design and Evolution of Protege-2000," SMI Technical Report, SMI-1999-0801, Stanford University, NY, USA, 1999.

[6] L.Y. Hsu, M.Y. Wu, and I.C. Chang, "Construction an Ontology-based Freeware Knowledge Sharing Web Platform with OWL," *Proc. of 2005 International Conference on Open Source*, Taipei Taiwan, 2005.

[7] Y.C. Lo, *The Art of Java*, GrandTech Computer Graphic Systems Incorporation, 2003, pp. 6-1~6-66.

[8] F. Menczer, "ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery," *Proc. of the 14th International Conference on Machine Learning*, Nashville, Tennessee, USA, 1997, pp. 227-235.

[9] S.J. Xing, "An Ontology-Based Adaptive Topical Crawling Algorithm," *Proc. of the 4th International Conference on Wireless Communications, Networking and Mobile Computing*, Dalian, China, 2008, pp. 1-4.

[10] S.Y. Yang, "A Website Model-Supported Focused Crawler for Search Agents," *Proc. of the 9th Joint*

Conference on Information Sciences, Kaohsiung, Taiwan, 2006, pp. 755-758.

[11] S.Y. Yang, F.C. Chuang, and C.S. Ho, "Ontology-Supported FAQ Processing and Ranking Techniques," *Journal of Intelligent Information Systems*, 28(3), 2007, pp. 233-251.

[12] S.Y. Yang, C.L. Hsu, D.L. Lee, and Lawrence Y. Deng, "FAQ-master: An Ontological Multi-Agent System for Web FAQ Services," *WSEAS Transactions on Information Science and Applications*, 5(3), 2008, 221-228.

[13] S.Y. Yang and C.L. Hsu, "Ontology-Supported Web Crawler for Information Integration on Call For Papers," *Proc. of 2009 International Conference on Machine Learning and Cybernetics*, Hebei, China, 2009, pp. 3354-3360.

[14] S.Y. Yang and C.L. Hsu, "An Ontological Proxy Agent with Prediction, CBR, and RBR Techniques for Fast Query Processing," *Expert Systems with Applications*, 36(5), 2009, pp. 9358-9370.

[15] S.Y. Yang and C.L. Hsu, "Ontology-Supported Web Recommender for Scholar Information," *Proc. of The Ninth International Conference on Hybrid Intelligent System*, ShenYang, China, 2009, pp. 271-276.