Ontology-Supported Portal Architecture for Scholar’s Webpages

Sheng-Yuan Yang¹ Chun-Liang Hsu²
¹Dept. of Computer and Communication Engineering, St. John’s University
²Dept. of Electrical Engineering, St. John’s University
Email:¹ysy, ²liangj@mail.sju.edu.tw

Abstract

This paper proposed an ontology-supported portal architecture: OntoPortal, which can integrate both internal components ontology-supported crawler and classifier to concern both mechanisms of precise searching and fast query of portals. The preliminarily experimental outcomes proved the technology proposed in this paper to be able to really up-rise the precision, recall rates of webpage searching, achieve the regular-level outcomes of webpage classification, and accordingly proved the feasibility of this architecture.

1 Introduction

In this most quickly universal era of Internet technology, the information of Internet does geometric-progressively increase. How to search advantage information to meet user queries in the information torrent of Internet has become the goal of lots of scholars to make efforts in all the time. Modern famous search engines, such as Google, which adapted query method with keywords and provided more than 8 billions index URLs [12] resulted in a hard-breathing searching outcome even with less keyword. 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 while 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 [5]. Hence, this paper proposed an ontology-supported Web Crawler, namely, OntoCrawler, to effectively filter noisy webpages and enhance the dominant of data searching.

Yahoo is a well-known webpages index structure, which can not only provide a fast way to get demand information of users, but also enhance the capability of keyword-based information searching described before. Therefore, this paper employed an ontology-supported classifier, namely, OntoClassifier, to classify those webpages collected by OntoCrawler so as to conveniently and fast provide necessary information to users. Those adopted techniques include such as normalization, segmentation, stop word filtering, and word stemming. Finally, to employ the term frequency as the base of classification, which is used to classify webpages with TF/IDF (Term Frequency / Inverse Document Frequency) [2,7,8].

Fig. 1 illustrates an OntoPortal architecture which combines with OntoCrawler and OntoClassifier described before. The system first checks Classified DB for any possible related solutions when the user enters query keywords by Interface. If anyone existed, the system would directly output them as the solutions to the user through Interface. If none existed, then it would invoke OntoCrawler to get related webpages by a search engine Google or Yahoo. On the one hand, the system straightforward presents those returned webpages to the user by Interface; on the other hand, it can proceed webpages classification by OntoClassifier and storage in Classified DB to strengthen the robustness of the database.

To sum up, this paper proposed an ontology-supported portal architecture: OntoPortal, which can integrate both internal components ontology-supported crawler and classifier to concern both mechanisms of precise searching and fast query of portals. The preliminarily experimental outcomes proved the technology proposed in this paper to be able to really up-rise the precision and recall rates of webpage searching, achieve the regular-level outcomes of webpage classification, and accordingly proved the feasibility of this architecture. The Scholar webpages domain is chosen as the target application of the proposed system and will be explained in the remaining sections.

2 Background and Developing Techniques

2.1 Ontology

Ontology was one theory in philosophy and primarily to explore knowledge of life and real objects; in artificial intelligent field it was used to define the content of domain knowledge, expressing knowledge, solve communication problems, and commonly share problems; in information technology field it offered much assistant for research and development of E-commerce and Knowledge Management [6,16,17]. Ontology provides complete semantic models, in which dedicated specified domain with related entities, attributes and basic knowledge among entities and all models have characteristics of sharing and reusing which could be applied for solving the problems of common sharing and communication. 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, so, ontology is a powerful tool to construct and maintain an information system [15,16].

2.2 Regular Expression

Regular expression is a character queue to describe specified objects. 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 [6]. For the operation of OntoCrawler [18], 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. However, for the operation of OntoClassifier [19], the regular expression meant the system removed the unmeaning words, including continuous blank spaces, line feed, Tab character, punctuation marks, and so on. In classification with above processing the precision rate of classification from being classified documents would be up-risen.

2.3 Processing of Classification

There are many classifiers combined with classification methodology such as TF (Term Frequency) and IDF (Inverse Document Frequency) [2,7,8], in which processed classification depended on term-frequency and selected feature-terms with characteristics to symbolize its document feature. The conception of TF was first proposed by Salton and McGill [11] while IDF was proposed by Spark Jones [13]. The reasons why they proposed the two methodologies was that the importance of every term appearing in the document was not quite the same, most of all, the importance was not necessarily the same even appearing in different articles. Therefore, combining the two methodologies could measure the importance of the feature terms.

Traditional statistics classifier must be accompanied with proper methodology of extracting feature and fetch out the proper features from training web-pages so as to gain the precision of classification. Hence, the quality of set of feature-terms would decide the classification precision. From the related literature, there were many resolutions such as [9], an extracting mechanism with squared relationship between constancy and Entropy could extract valid feature-terms from training web-pages in extremely short time without human assistance. Tan [14] especially proposed one fairer feature-select method, in which all kinds of document could be treated fairly and the input dimension could be decreased more. Besides, a practical division for articles with adjustable fuzzy learning of web-net structure was proposed to achieve the superiority of learning the knowledge of experts’ classification; furthermore, to get to speed classification. From those theories above, the conception of either taking advantage of information theory or machine learning was inclining to use symbols (or numbers) calculation. The processes and its outcomes are difficult to be understood. In this paper we adapted domain ontology to support the processing of classification that not only arise the classifier’s efficiency but also make the processing and outcomes of the classification easily understood.

2.4 Related Developing Techniques

This system adapted MS SQL Server as backend knowledge-database sharing platform based on ontology. MS SQL Server is one mostly popular relational database management system. SQL (Structured Query Language) is one query language to get the data in the database. The system oneself was developed with Java [10].

The construction tool of ontology, Protégé [4], was an ontology freeware developed by SMI (Stanford Medical Informatics). Protégé not only was one of the most important platforms to construct ontology but also the most frequently adapted one. Protégé 3.3.1 was adapted in this paper, which is a construction tool of ontology with open-used, free-charged, and well-defined features. The most special feature is that its framework is constructed by ontology concepts. It used multi components to edit and make ontology and led knowledge workers to constructing knowledge managed 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.

3 Ontology Construction and System Architecture

3.1 Construction of Ontology Database

Our ontology is a knowledge sharing database which was constructed for specific domain. That is to say it could take advantage of built ontology database of some scholars to support OntoCrawler for querying webpage of related scholars and assist OntoClassifier in processing of webpage classification. In the mentioned ontology database, it included two constructed stages; one is statistics and analysis of related concepts of scholars, the other is construction of ontology database. We detailed the procedures as below.

First of all, we conducted statistics and survey of homepage of related scholars to fetch out the related concepts and their synonym appearing in the homepage. In Fig. 2, illustrated statistics and analysis of 50 datum about related concepts from webpage of scholars in National Taiwan University. The symbol “○” stood for this concept definitely appearing in the homepage; the symbol “□” stood for the concept appearing the next hyperlink-page. Fig. 3 indicated the structure of domain ontology of scholars in Protégé, taking the middle frame of the screen for instance, the related concept
“Education” was linking behind with “M.S”, “PH.D”, and “B.S”. In application, we defined those as related concepts and that means “education” is nothing but an combination of these related concepts that would be conveniently interpreted by OntoCrawler to compare with content of the queried webpage, and if the compared outcomes were corresponding to any item among the four, we would infer the related concept “Education” as matched condition for web page querying.

Fig. 3 The ontology structure of Scholars

Fig. 4 The ontology structure of Scholars

Fig. 4 shows the second stage of ontology constructing of Scholars, in which the main part work is to transfer the ontology built with Protégé into MS SQL database. The procedures are as following:

1. Exporting an XML file constructed with Protégé knowledge base and then importing into MS Excel for correcting. That was the strong evidence of knowledge reusing and fast embedding within Protégé.
2. Finally importing MS Excel into MS SQL Server to finish the ontology construction of this system.

3.2 OntoCrawler

Fig. 5 showed the operation system structure of OntoCrawler, and related techniques and functions of every part were described as below.

1. **Action**: to transfer internal query into URI code, and then embed into Google’s query URL: an example as follow

   http://www.google.com.tw/search?hl=zh-TW&q=%E5%AD%B8%E8%80%85&meta=

2. **LinkToGoogle**: to declare an URL object and add Google 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**: to use regular expression to search for whether there are matched URL. But it couldn’t retrieve all the linkages in one time out because of the Google webpage editing with indenting. So we used a “for” loop and ran for twice. The semantic of the two in regular expression were slightly different so as to completely fetch out related hyperlinks corresponding to the conditions. Finally, added all hyperlinks on String variable “RetrieveLink” and output the txt file to provide the system for further processing.

4. **RetrieveContent**: to use BufferedReader (Java Class) 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 added on String variable s3 and output it as text file so as to let system conduct further processing. After completing all procedures mentioned above, we could use SearchMatches method (described later) to judge whether the webpage was located in the range we hoped to query; supposed the answer was “yes”, we would execute RemoveHTMLLabel (described later) 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. Remember to clear RetrieveLink lest the next query should cause errors.

5. **SearchMatches**: supporting RetrieveContent internal calling service to judge whether the webpage was the range we queried. It linked the ontology database and fetched out the content to compare content of s3 when using this linking method. If there were any return value corresponding to the value we set, and then system would return one “true” Boolean variable “matches.” That meant the webpage matched our query condition, on the other hand, if returned “false” meant the webpage didn’t match our query condition.
3.3 OntoClassifier

![Fig. 6 System operation flowchart of OntoClassifier](image)

Fig. 6 illustrated the system operational structure and flowchart of OntoClassifier. The Source Text originated from webpage crawler developed by us with support of ontology - OntoCrawler which could highly arise the Precision Rate of webpage crawler and Recall Rate up to more than 90% [18]. The rest functions and related techniques were described as below.

(1) Action: input file format was *.txt that was pure text of webpage after excluding labels. Because there quite a lot difference in grammar structure and semantic expression between Chinese and English, it must managed to respectively. For example, the continuous space, Tab, \n, and \t characters were used to division or beautify the content and have nothing to do with semantic expression. But, in English, the division of phrases needs space and before dividing phases the non-semantic characters mentioned above must be filtered. The initial task included as following: loading Stop Word database, database of ontology, formal database, and SourceText document for further processing.

(2) Text Processing:
   (a) Formalization: to load SourceText and formalize database array, and then search SourceText to replace every non-expression word with space character till the end of SourceText.
   (b) Segmentation: to divide SourceText into the least semantic unit - vocabulary. Searching every space character in the document from the first character until the next one, and this is what we defined single phrase. Then system extracted every single phrase and stored into array FirstArray [3].
   (c) Filter Stop words: stop words or noise words were words that often appeared in document but have no contribution for reading whole document such as “is”, “of”, and “the” etc. we used “stop list” to store those words and ought to be excluded when indexing words and phrases so as to decrease the noise in document and increase classification precision [15].
   (d) Stemming: stem transformation means transfer different word type into stem for example, “connection”, “connections”, “connective”, “connected”, “connecting” would be all transferred as “connect” to arise the extracting precision [15]. In practical processing, we could use Java API JWNLP provided by WordNet (http://jwordnet.sourceforge.net/) to accomplish it.

(3) TF Statistics: to build up one terms-frequency array corresponding to vocabulary array from the first vocabulary as the bas. After comparing with every vocabulary, if one vocabulary appeared repeatedly, then we add one recode to the tem-frequency. Then delete the repeated term and tem-frequency record till the last vocabulary.

(4) Filter and ordering: to order the term-frequency mentioned above with decrement to enhance the readability of data. To speed the managed process and lead into Ontology, we divided the steps into two stages. Stage 1: to order every data with decrement according to term-frequency, filter out every space data, and finally record total amount of records. Stage 2: to do the processing of classification. First, to load into Ontology database array, to gain feature terms which has threshold value more than predefined threshold according to the ratio between feature term and total number of terms, and then compare the feature terms with the coverage rate in Support.txt array. If the value were higher than threshold, then we add “*” onto in front of the file name of the document appearing the output screen and all marked with “*” would be classified as the same kind.

4 System Verification and Comparison

The evaluation of the overall performance of our proposed system involves lots of manpower and time-consuming. Here, we first focus on the performance evaluation of the most important modules, for instance, OntoCrawler and OntoClassifier of the OntoPortal architecture. Our philosophy is that if OntoCrawler can precisely collect domain related webpages, and then OntoClassifier can do the best processing of classification, and finally we not only can precisely and effectively produce solutions to users, but also can efficiently improve the quality of the retrieved to meet the user requests, which would be clear and detailed from system demonstrations.

4.1 Verification of OntoCrawler

In this experiment, we compared our query technique with the most popular query engines Google and Yahoo. We used five different keywords as webpage query of related information. First of all, we keyed in the same set of keywords in Google and Yahoo. The comparison result was shown in Table 1, in which NW means the number of total returned webpages; NWF means number of correct returned webpages; NWG means number of related returned webpages but they were not necessarily the correct webpage. Here, we used equations (1) and (2)
4.2 Verification of OntoClassifier

Table 3 Classification outcome of 4 Experiment webpage

<table>
<thead>
<tr>
<th>Experiment</th>
<th>NW</th>
<th>NW - NW</th>
<th>R_P</th>
<th>R_R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>92</td>
<td>92</td>
<td>92%</td>
<td>92%</td>
</tr>
<tr>
<td>2</td>
<td>92</td>
<td>92</td>
<td>92%</td>
<td>92%</td>
</tr>
<tr>
<td>3</td>
<td>92</td>
<td>92</td>
<td>92%</td>
<td>92%</td>
</tr>
<tr>
<td>4</td>
<td>92</td>
<td>92</td>
<td>92%</td>
<td>92%</td>
</tr>
</tbody>
</table>

In this experiment, we fetched 32 data of scholar webpage from OntoCrawler database and every group 8 data was composed of 8 data so that we got 4 groups totally. First of all, we took webpage of “Yang, Sheng-Yuan” - a teacher working in Dept. of Computer and Communication Engineering in St. John’s University -- as target to analyze its content and extract some terms with feather, and then to analyze the webpage of mentioned 4 groups. The experiment outcome was shown in Table 3. N_{SD} stands for Total Number of System Discrimination; N_{SM} stands for Total Number of System Miss that means this document would be judged as the same classification even it was not. N_{EC} stands for Total Number of Expert Check. Finally, we used equation (3) to define Precision Rate of Classification (P_C). The average P_C of 4 times classification was around 92%, and this show the technique was valid and available.

\[ P_C = \frac{N_{EC}}{N_{SD} + N_{SM} + N_{EC}} \]  

5 Conclusions and Future Works

We had proposed an ontology-supported portal architecture: OntoPortal, which can integrate both internal components ontology-supported crawler (OntoCrawler) and classifier (OntoClassifier) to concern both mechanisms of precise searching and fast query of portals. First, OntoCrawler has been practically applied on Google and Yahoo searching engines to actively search for webpages of different scholars and stored the searching outcomes into databases for future management of back-end systems. The preliminarily experimental outcomes indicated that this technique could definitely up-rise precision and recall rate of webpage query up to around 90%. Furthermore, under the domain ontology-supported; OntoClassifier can handle and analyze those document contents for actively extracting and computing the domain word frequency to process documents classification of related scholars. The experiment outcomes proved the technology could really up-rise the precision rate of webpage classification up to 92%. In short, synthesizing above experimental outcomes, the techniques proposed in this paper could definitely up-rise precision and recall rate of webpage query and achieve a regular precision rate of webpage classification. Those evidences could confirm this architecture’s availability.

Inside Fig. 1, the Interface module has not been developed, which is our first work in the future. We will plan to transplant the template-based linguistic processing technique from our previous Interface Agent.
[20], in which users’ intention and focus degree of the user queries can be correctly understood by the system up to eighty percent, and accordingly provides the query solutions with higher satisfaction. Secondly, OntoCrawler could easily combine with other search engines by clarifying the meaning of related query sentences/words, and then a query base could be easily achieved. 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. Finally, on the OntoClassifier hand, analyzing the contents of webpage in initial experiment 4 (Table 3), the reason why the precision of classification was only at 67% and its worse effect under the ontology supported using TFIDF to calculate the weight of categorization was due to imperfection of ontology itself. Maybe how to construct a proper database of ontology would be the critical research direction. Next, in present system was lack of automatic Stemming, and we plan to employ Java API JWNL (http://jwordnet.sourceforge.net/) provided by WordNet to accomplish those related steps. Finally, it would be another critical challenge how to upgrade the management capability for webpage in system especially for Chinese webpage.

Acknowledgements

The authors would like to thank T.A. Chen, C.F. Wu, H.C. Chiang, and C.S. Wu for their assistance in system implementation and experiments. This work was supported by the National Science Council, R.O.C., under Grant NSC-95-2221-E-129-019.

References