An Information Extraction Engine for Web Discussion Forums

Hanny Yulius Limanto, Nguyen Ngoc Giang, Vo Tan Trung,
Nguyen Quang Huy, Jun Zhang, Qi He
Nanyang Technological University
Nanyang Avenue, Singapore 639798
(65) 6791-1744
{H943701, A0227955A, 023448171, AS0225608, JZhang, QIHE0001}@ntu.edu.sg

ABSTRACT

In this poster, we present an information extraction engine for web-based forums. The engine analyzes the HTML files crawled from web forums, deduces the wrapper (template) of the pages and extracts the information about posts (e.g., author, title, content, number of replies and views, etc.). Extraction is an important module for forum search engine, since it helps to understand the content of a forum HTML page and facilitates ranking during retrieval. We discuss the system architecture of the extraction engine in the context of a forum search engine and present various components in the extraction engine. We also introduce briefly the extraction process and discuss some implementation issues.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing - *abstracting methods*.

General Terms

Algorithms.

Keywords

Information Extraction, Information Retrieval, Search Engine, Discussion Board, Forums.

1. INTRODUCTION

Web discussion forums are web-based discussion boards where people exchange ideas by posting articles and replies. Forum is one of the major communication platforms for Internet users with similar interests. Unlike newsgroups, web forums provide diverse functionalities, nice user interface and easy usage. As the emergence of numerous web forums, a huge amount of new and real-time information is generated and this calls for an effective search support. The forum software provides basic search functions restricted to only one forum. General search engines support searching across forums provided these forums can be reached by their crawler. However, regardless whether they are of general purpose, e.g., Google [3], or specifically designed for web forums, e.g., Lycos Discussions [4], they index and rank each forum page as a whole without the knowledge of the content inside the pages. Combined with the fact that link analysis also fails, their ranking is not effective and advanced searching (e.g., search based on title) is not supported.

Motivated by this, in this poster, we present an information extraction engine for web forums that automatically discovers the structures of the forum pages and extracts information inside the pages. The idea of information extraction has been previously proposed in various papers [1, 2, 5]. The extracted content can be further indexed to support effective retrieval. Sophisticated post-based (instead of page-based as in most of the search engines nowadays) ranking functions can be developed utilizing the extract content, e.g., it is possible to give a high rank to a post with large number of views (or replies).

The proposed system architecture and algorithms in this poster can be generalized and applied to any web content that is generated by templates. In this poster, however, we will focus on the web forum application. We first describe the architecture of the extraction engine and study how extraction fits in the forum search engine big picture. Then we elaborate on each component of the extraction engine and present briefly the algorithms. Finally, we conclude the contribution of the poster.

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

2. SYSTEM ARCHITECTURE

In order to present a complete picture, we depict the system architecture of a forum search engine in Figure 1, with the information extraction engine highlighted. The whole system is a web application that allows users to search for information of their interest over all the web forums. The major modules of the forum search engine are crawler, wrapper generator, extractor and query processor. Together with the wrapper database, extractor and wrapper generator constitutes the extraction engine.

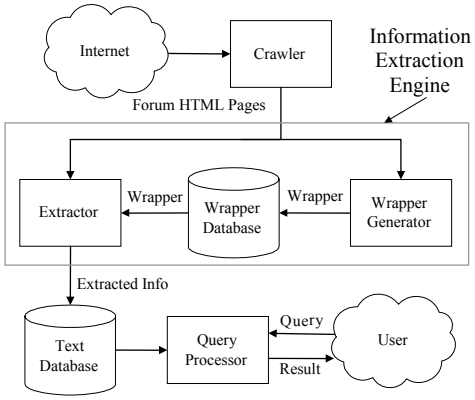

Figure 1. Forum Search Engine System Architecture

Crawler continuously discovers and retrieves HTML pages from the Internet. For each of the pages obtained, crawler utilizes a trained classifier to determine whether it is a forum page. Forum pages are passed to information extraction engine, which consists of wrapper generator and extractor. Wrapper generator studies an HTML pages and discovers the repeated patterns in the page. A wrapper (represented as a regular expression) is deduced from the

page, and stored in wrapper database. Later when a page with the similar URL prefix arrives, we directly retrieve the wrapper for the page from the database without running the wrapper generator again. Extractor extracts information by matching the HTML page with the corresponding wrapper. The extracted information is directed to the text database, where index is built to support efficient retrieval. Query processor accepts queries from the user, retrieves relevant information from the text database, ranks the result accordingly based on the ranking function specifically defined for web forums and returns it to the user.

3. EXTRACTION PROCESS

The extraction is performed in two steps: i) deduce the wrapper of the page (Session 3.1); ii) extract information inside the page using the generated wrapper (Session 3.2).

3.1 Wrapper Generation

The wrapper generator produces the wrapper of an HTML page by analyzing the page content. It is a reverse engineering process to obtain the original template of the page. The wrapper generation algorithm used in the system is an improved version of the algorithm presented in [5]. Firstly, the HTML page is cleaned to adhere to the XHTML specification. The XHTML page is then tokenized by treating HTML tags and text between tags as tokens. A token suffix tree is built on these tokens to discover continuously repeated patterns. A continuously repeated pattern is a token set that repeats itself consecutively in a token string. Figure 2 illustrates a token string and the corresponding suffix tree. As an improvement of the algorithm in [5], we only insert substrings starting with an open tag into the suffix tree, since a meaningful repetition always begins with an open tag.

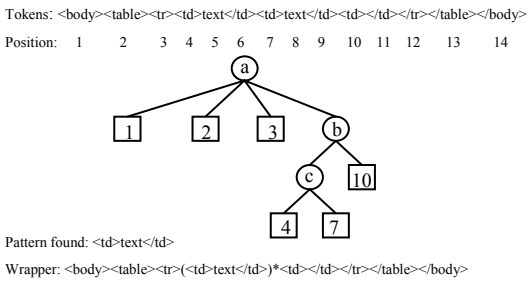

Figure 2. Token Suffix Tree

To detect repeated patterns, we traverse the suffix tree and for each non-leaf node, we check all its child nodes and see whether consecutive strings can be found. In the suffix tree of Figure 2, under node *c*, we find substring *4* and *7*. They both start with *<td>text</td>*, and they are consecutive. Thus they constitute a continuously repeated pattern. We add an additional step to the original algorithm, filtering out the patterns without *text* or unbalanced (i.e., open and close tags do not match). Among all the patterns found in one iteration, we select the shortest one and build a new suffix tree by replacing it to its collapsed form. The same process is repeated until no new patterns can be found.

The above algorithm finds repetition on the HTML tag level only. It cannot handle cases where text phrase is part of the repetition. Figure 3 shows an example where text *Registered:* is in fact from the template and appears repetitively. In order to find such repetition, we propose the following algorithm: we scan all *text* token *t* that is part of a repeated structure. If we find some common word *w* appearing constantly in every instance of *t*, we treat *w* as a text repetition. In the example of Figure 3, we find that *Registered:* appears once in each of the text token, so it is regarded as part of the template.

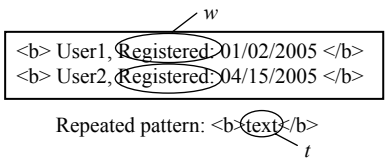

Figure 3. An Example of Text in the Template

For optional element detection, we employ string alignment algorithm. Observe that optional element always appears inside repetitions, we only need to consider each repetitive pattern and compare it to the tokens adjacent to it. If the tokens are similar, we can conclude that the differences between the two patterns are due to optional tokens.

3.2 Data Extraction

After the wrapper is generated, both the wrapper and the tidied XHTML page are sent to extractor. We utilize a non-deterministic finite-state automaton (NFA) to extract data. Extractor consists of two modules: i) converting wrapper into a NFA and ii) building a NFA engine to extract data. Figure 4 shows the structure of a data extractor.

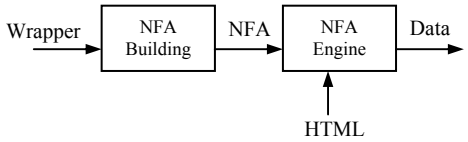

Figure 4. Structure of a Data Extractor

Thompson's algorithm is employed to convert wrapper (in form of regular expression) to a NFA. As an improvement, we reduce the number of states in the NFA by: i) treating *text* token as a single alphabet; ii) treating set of consecutive non-text tokens not interfered by regular operators (e.g., '*' and '|') as a single alphabet.

The constructed NFA is fed to an NFA engine, together with the XHTML page. NFA engine traverses the NFA and maps the source page with the wrapper by exploiting all possible ways of matching. When a match is detected, the NFA engine backtracks. When a *text* alphabet is encountered during backtracking, we perform extraction of data.

4. CONCLUSION

In this poster, we propose a novel information extraction engine for web forums. The engine generates wrappers by studying the structure of the forum HTML files and extracts data inside the pages using the created wrappers. The extraction enables advanced searching and improves the effectiveness of ranking during retrieval.

5. REFERENCES

[1] Arasu, A. and Garcia-Molina, H. Extracting structured data from web pages. SIGMOD 2003, 337-348

[2] Crescenzi, V., Mecca G., and Merialdo P. ROADRUNNER: towards automatic data extraction from large web sites. VLDB 2001, 109-118

[3] Google: http://www.google.com

[4] Lycos Discussion: http://discussion.lycos.com

[5] Wang, J. and Lochovsky, F.H. Data extraction and label assignment for web databases. WWW 2003, 187-196