# Component search engine using tree-view interface for tourist blogs

Zeng, Jun
Graduate School of Information Science, Kyushu University

Hirokawa, Sachio
Research Institute for Information Technology, Kyushu University



出版情報：Proceedings of 2011 3rd International Conference on Awareness Science and Technology, iCAST 2011, pp.331-335, 2011-12-01
バージョン：
権利関係：

# Component Search Engine using Tree-View Interface for Tourist Blogs

Jun Zeng
Graduate School of Information Science,
Kyushu University
Fukuoka, Japan
sousyun_kyusyu@yahoo.co.jp

Sachio Hirokawa
Research Institute for Information Technology,
Kyushu University
Fukuoka, Japan
hirokawa@cc.kyushu-u.ac.jp

***Abstract*— The existing search engines return the whole web pages as the search results, which make user spend extra time to read the useless information before finding the information they really want. We propose a novel search engine model called "Component Search Engine", which can return the contents satisfying user's query rather than the whole pages. For achieving the purpose, we adopt a Tree-View interface to display the results. Through usability study, we determinate that Component Search Engine using Tree-View interface can improve user's searching experience and efficiency.**

***Keywords- Component; Search Engine; Tree-View; Tourism Blog***

## I. INTRODUCTION

The existing search engines, such as google, yahoo and baidu, always return the whole web pages as the search results, which is not easy for uses to find out the useful information at a glance. That's because the web pages often contain many irrelevant contents such as advertisement, navigation, menu and so on, which are called as "noise". There are already many researches on efficient content extraction from web pages. Perhaps, it will get satisfying results, if the pages have a uniform template (for instance, the news web pages). Nevertheless, there are massive web pages which have personalized templates instead of a uniform template, such as Wiki and Blogs. It's difficult to extract the information that the users really want, even though the noise has been removed. It is because that the Wiki and Blog pages may contain several main contents that may be irrelevant to each other. For example, there are three articles A, B and C in one blog page, which have different contents (A is an editorial comment, B is a diary and C is a travel note about Japan). Consider the query "where is interesting in Japan?". Assume that only article C can satisfy the user's query. Obviously, all of the three articles will be extracted as main contents by the existing information extraction approaches. However, in order to find out the article C, the user has to spend extra time in reading article A and B, which is useless for the user. In this case, article A and B may be considered as noise by the user. Moreover, in contrast to advertisement and navigation, perhaps this kind of noise is more indistinguishable. It's almost impossible to adapt for different users' needs in any case by existing information extraction approaches.

To solve problem mentioned above, in this paper we propose a novel search engine model called "Component Search Engine", in which the search results are not the whole web pages any more, instead, only the contents satisfying users' query in the page will be returned as search results. For achieving the purpose, we adopt a Tree-View Interface to display the search results, which can help users to acquire the information needed easily and efficiently.

The authors of this paper have been working on novel search engine models [1], [2]. The tourism blogs of site "Kyushu seifuku Blog"[1] were chosen to analysis the effectiveness of the models. We collected 1,303 blog entities and saved as html files and extracted 136,368 components from these blog.

## II. RELATED WORKS

There are two kinds of search engines – a vector space model based keywords search engine and semi-structured documents search engine. The vector space model is an algebraic model for representing text documents as vectors of identifier. With the increasing of HTML/XML documents, semi-structured documents search engine has gained more attention. Moreover, it has been an issue to integrate the two kinds search engine. Therefore, there is a large body of related work in content identification and information retrieval from HTML/XML documents.

Gupta et al. [3] developed a DOM tree based framework that employs easily extensible set of techniques that incorporate advantages of previous work on content extraction. Cai-Nicolas et al. [4] presented an approach to extract content from news Web pages in an unsupervised fashion, which is based on distilling linguistic and structural features from text blocks in HTML pages. Lei et al. [5] presented a method which combines webpage layout analysis with DOM tree rule-base method.

However all of these papers just presented how to extract the contents effectively and accurately, rather than consider the users' needs. Because the useful contents are relative, it will change with change of users' needs. Therefore, we won't try to extract the useful contents, but score the contents. According to the score, the contents that satisfy user's need will get high rankings among the search results.

[1]http://www.welcomekyushu.jp/

As the prior study of Component Search Engine, Hirokawa et al. [6] proposed a component-based search engine in which the content components gain a high score in the search results. Moreover he ranked the search results according the score of each page. However, he didn't realize the real component search, because the unit of search results was still web page, not the components.

## III. COMPONENT SEARCH ENGINE

### A. Definition of Component

A web page can be divided into many blocks. In this paper, we call these blocks as "components". Figure 1 shows a blog page. This page contains Logo (A), Navigation (B), Article (C), Search Box (F) and Advertisement (G). Also, (C) contains Title (D) and Text Area (E). We call A~F as components of the blog page. Some components can be divided into smaller components, for instance, C can be divided into D and E. In this case, we call D and E as "sub-components" of C. Fig. 2 shows the HTML-Tree architecture of the blog page. Each component has a corresponding HTML-tag and each HTML-tag has a unique X-path [7]. For example, the X-path of component D is "html/body/div[3]/div". So a component can be located by a URL of the web page and an X-path of HTML-tag. Therefore, we define a component of a web page as follows:

$$Comp_i = \{(U, P_i) | i \in [1, I]\} \quad (1)$$

Where $U$ is the URL of the page, $P_i$ is the X-path of $i$th HTML-tag, $I$ is the total number of HTML-tags in the page. If $Comp_i$ is a sub-component of $Comp_j$, we define

$$Comp_i \subset Comp_j \quad i, j \in [1, I] \quad (2)$$


Figure 1. A Blog Page

### B. Component Index

In order to realize the Component Search Engine, we need to build a component index that is different from page index used by existing search engines. We pay our attention to the leaves of HTML-trees which do not have any sub-components. We call this kind of component as "leaf component". It can guarantee that the content of each component is unique. Therefore, we divide a web page into leaves of HTML-tree and build the component index. Figure 3 shows a fragment of component index.

The lines beginning with "@" present the identifier of a component. For example, "@ 1-12" means the 12th leaf component of the page whose id is "1". The other lines present the index keywords and their frequencies. The keyword beginning with "h:" is the id of the HTML file, and the keyword beginning with "p:" is the X-path of component.

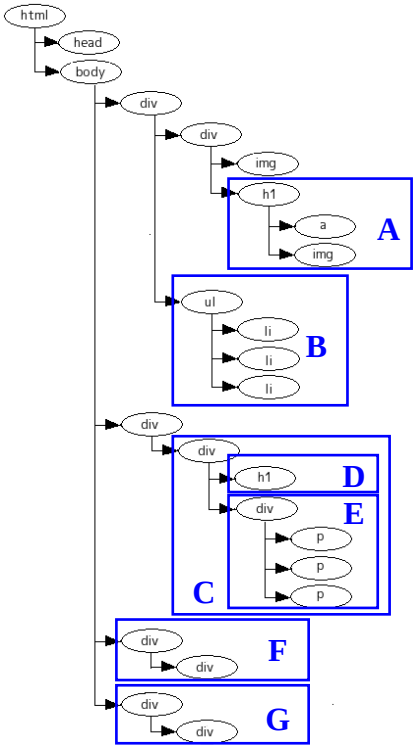


Figure 2. HTML Tree

```
…
@ 1-12
1 h:1
1 p:/html/body/div[2]/div/div/p[1]/
1 what
1 can
1 one
1 …
@ 1-13
1 h:1
1 p:/html/body/div[2]/div/div/p[2]/
1 most
1 hotel
1….
:
:
```

Figure 3. Fragment of Component Index

### C. Ranking Search Results

The usefulness of a search engine depends on the relevance of the result set it gives back. While there may be a large number of web pages that include a particular word or phrase,

some pages may be more relevant, popular, or authoritative than others. Most search engines employ methods to rank the results to provide the "best" results first. Therefore, ranking the search results is a necessary step for developing a web search engine. The existing search engines just sort the web pages through some algorithm (for instance, PageRank of Google). However, in Component Search Engine, the component is the unit of search results, so we must sort the components as well. Therefore, we divide the task into two steps – ranking pages and ranking components. Figure 4 shows the process of ranking the search results.

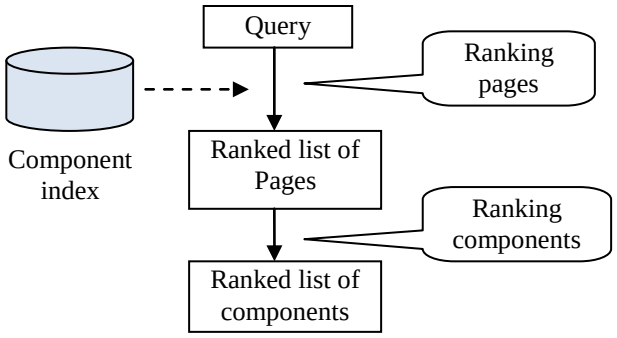


Figure 4. Fragment of Component Index

*1) Ranking pages:* For sorting the pages, we introduce a formula for calculating the score of pages as follows[6]:

$$Score(P) = \sum_{i=1}^{N} NW(C_i) * Depth(C_i) \;\; i \in [1, N] \qquad (3)$$

Where $C_i$ is the *i*th leaf component in page *P*, NW($C_i$) is the number of the distinct words in component $C_i$, $Depth(C_i)$ is the length of X-path of component $C_i$, and *N* is the total number of leaf components in page *P*. For example, if the X-path of component $C_i$ is "/html/body/div[2]/div/div/p[1]/", then $Depth(C_i)$ will be 6. If a page gains a higher score, it will appear in a higher ranking.

*2) Ranking components:* We introduce another formula to calculating the score of each component in page *P* as follows:

$$Score(C_k) = ln(Depth(C_k) + 1) * NK(C_k) \;\; k \in [1, K] \qquad (4)$$

The formula is for calculating all the components in page *P*, so $C_k$ is the *k*th component (not only the leaf component) and *K* is the total number of components in page *P*. $Depth(C_k)$ is the length of X-path of component $C_k$. If component $C_k$ is a leaf component, $NK(C_k)$ will be "1" when $C_k$ contains keyword, otherwise $NK(C_k)$ will be "0". If $C_k$ is not a leaf component, $NK(C_k)$ will be the sum of leaf components $C_j$ where $C_j \subset C_k$ and $NK(C_j)$ =1. If a component gains a higher score, it will appear in a higher ranking.

The formula (4) is based on the hypothesis that the component which contains the keyword is more likely to contain the content that can satisfy user's need. In this case, the root component may be considered as the most important component because it contains all of the contents, but it is against the purpose of component search. So we add $ln(Depth(C_k) + 1)$ to avoid this case. Consequently, only the useful components can gain high scores and appear in high ranking.

## IV. User Interface Design

### A. Design Goal

The goal of component search is to help user to get the useful information at a glance. This is also the goal of user interface design. As the principles, interface design should organize the user interface purposefully, in meaningful and useful ways. Therefore, we will adopt the interface form that is easy to understand as far as possible. We divide the user interface into two stages: search result list and detail pages. In this section, we use the blog data mentioned in Section I to introduce the interface of component search engine.

### B. Search Result List

First, we enter a query by using the keyword "麺" (noodle). Figure 5 shows interface of the result list of the query. This page displays the top ten results, but here we only show the top three of them because of the page limit. Each result contains two parts: title and component. The "Title" displays the title of each blog. When user clicks the link of each title, the detail page will be opened. The "Component" displays the top component of each blog through the ranking algorithm mentioned in Section III. Generally, the search result list page is similar with the existing search engines such as google, bing etc. That's because users have become accustomed to this way for displaying the result list.


Figure 5. Search Result List

### C. Detail Pages using Tree-View Interface

Tree-View is an intuitive and common form of user interface. Through ordinary users may do not known about HTML-tree architecture, however, they may be no stranger to Tree-View, since they may use folder tree of windows OS everyday. Moreover, Tree-View makes it possible to organize the information in the groups formed with related elements. [8] Therefore, it is very suitable for the Component Search Engine.

Figure 6 shows an example of the detail page. In contrast to the traditional search engines, the detail page contains three parts, rather than just displays the original page. Part ① is the original page of search result. Part ② displays the Tree-View of HTML-tree of the page on the left. If the node contains keyword there will be a "smile face" on the node, which can help user to find the sentences containing keyword quickly. Part ③ displays the contents of the top ranking component. Initially, only the node that corresponds to the top ranking component is open, and the other nodes are closed. User can click any node he likes, and the contents that the node contains will be displayed in part ③. Also, user can click the "default" button to display the top ranking component again.

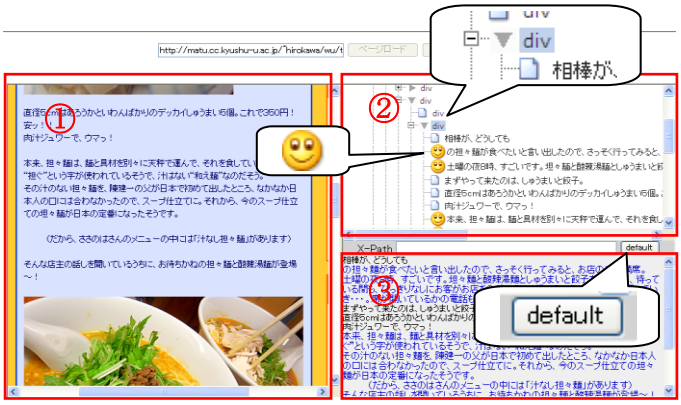


Figure 6. Detail Page Using Tree-View Interface

## V. USABLITY STUDY

### A. Methodology and Procedure

To evaluate the usability of Component Search Engine, we introduce a usability experiment. We developed two search engines, one is Component Search Engine (C-Search), and the other one is a Normal Search Engine (N-Search) without component ranking and Tree-View used detail page, also the latter one don't display the useful component in the search results. We collected 1,303 tourism blogs from the site "Kyushu seifuku Blog" as experimental data. 10 participants joined in the experiment. Before the experiment, we introduced participants the features of Component Search Engine, since every one has not used the Component Search Engine before. We design the task of usability experiment as follows:

- First, participants use C-Search and N-Search to search the information. They must choose tourism-related keywords, since the experiment data are tourism blogs.
- Second, participants browsing the detail pages of search results. They must find and save all the sentences containing the keyword. Every participant must browse 10 pages: C-Search 5 pages and N-Search 5 pages.

And then we recorded the following information:

1) *URL:* URLs of blogs that participants browsed;

2) *Keyword:* Keywords that participants chose;

3) *Sentence:* Sentences that participants found and saved

4) *Time:* The time from participants begin to browse a detail page to they found out all of the sentences.

### B. Results and Analysis

We got 100 experiment records: C-Search 50 records and N-Search 50 records. The experiment is based on the hypothesis that the component which contains the keyword is more likely to contain the content that can satisfy user's need. Therefore, we ask the participants to find and save the sentences containing the keywords. By analyzing the time spent, we can evaluate if the Component Search Engine can improve user's searching experience and efficiency.

Before analyzing the time spent, it is necessary to judge whether the records are valid or not. That's because participants might be cursory when they were doing the experiment. They might not find out all the right sentences or just use Ctrl+A & Ctrl+C to copy all the sentences. Therefore, we introduce two parameters as follows:

$$R_1(P) = \frac{TS'}{TS} \quad (5)$$

$$R_2(P) = \frac{NS'}{TS'+NS'} \quad (6)$$

Where $TS$ is the total number of sentences that contain keyword in page $P$, $TS'$ is number of sentences that are found by participants and contain keyword in page $P$, $NS'$ is the number of sentences that are found by participants and don't contain keyword in page $P$. We consider that only the records that $R_1(P) > 0.8$ and $R_2(P) < 0.2$ are valid records. TABLE I shows the number of valid records in this experiment. Finally, we got 88 valid records: C-Search 46 and N-Search 42.

TABLE I. NUMBER OF VALID RECORDS

| | C-Search | N-Search |
|---|---|---|
| $R_1(P) > 0.8$ & $R_2(P) < 0.2$ | 46 | 42 |

TABLE II. TIME SPENT

| | C-Search | N-Search |
|---|---|---|
| Average Time | 47.5 sec | 98.2 sec |
| Min Time | 15 sec | 35 sec |
| Max Time | 85 sec | 103 sec |

TABLE II shows the time spent when participants tried to find the sentences containing keyword through C-Search and N-Search. It is obvious that participants have spent less time to find out the sentences containing keyword when used Component Search Engine. These results determinate that Component Search Engine can improve user's searching experience and efficiency.

## VI. Conclusion and Future Works

The existing search engines always return the whole web pages as the search results, which make user have to spend extra time to read the useless parts of the pages. In this paper, we proposed a novel search engine model called "Component Search Engine". In this search engine, the unit of search results is component, rather than the whole web page. For achieving the purpose, we not only rank the pages, but also rank the components. In order to make the search results easy to understand, we adopt a Tree-View Interface to display the detail pages of search results. The usability study determinates that Component Search Engine can help users to find the information needed easily and efficiently.

In the future we plan to analyze user's keyword to find out user's need. According the user's need, we can improve the ranking approach, so as to better meet the user's demand.

## VII. Acknowledgment

The authors appreciate many comments and discussion by the members of the laboratory. Particularly, we thank Prof. Itou, Prof. Nakatoh and Prof. Yin.

## References

[1] X. Wu, S. Hirokawa, C. Yin, T. Nakatoh, Y. Tabata, Extraction and Comparison of Tourism Information on the Web, Proc. AROB2011, 2011

[2] C. Yin, T. Nakatoh, S. Hirokawa, X. Wu, J. Zeng, A proposal of search engine "XYZ" for tourism events, Proc. JCAI2010, 2010

[3] Gupta, S., Kaiser, G., Neistadt, D. and Grimm, P., DOM-based Content Extraction of HTML Documents, in the proceedings of the 12th World Wide Web conference (WWW 2003), Budapest, Hungary, May 2003.

[4] Cai-Nicolas Ziegler, Michal Skubacz, Content Extraction from News Pages Using Particle Swarm Optimization on Linguistic and Structural Features, WI '07 Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 2007.

[5] Lei Fu, Yao Meng, Yingju Xia, Hao Yu, Web Content Extraction based on Webpage Layout Analysis, 2010 Second International Conference on Information Technology and Computer Science, July 24-25, 2010.

[6] Sachio Hirokawa, Chengjiu Yin, Tetsuya Nakatoh: Component-Based Search Engine for Blogs, Proc. FUZZ-IEEE2011, pp.1074--1078, 2011

[7] http://www.w3.org/TR/xpath/

[8] Nilsson, M., Palmér, M. & Brase, J.. The LOM RDF binding - Principles and implementation. In Proceedings of the 3rd Annual ARIADNE Conference, November 20-21, 2003.