

Component-Based Search Engine for Blogs

Sachio Hirokawa, Chengjiu Yin, Tetsuya Nakatoh

Research Institute for Information Technology

Kyushu University

Fukuoka 812-8581, JAPAN

Email: {hirokawa,yin,nakatoh}@cc.kyushu-u.ac.jp

Abstract—A wrapper is a program that selectively extracts a necessary part (component) from Web pages. Automatic or semi-automatic wrapper construction is crucial to achieve a fine grained search engine for Web pages. However, this is not an easy task to achieve. This paper proposes a component-based search engine in which the content components gain a high score in the search results. Thus, the required segments for a query can be obtained without using a wrapper.

Index Terms—search engine; blogs; ranking; semi-structured document

I. INTRODUCTION

Information extraction is a necessary step for developing a Web search engine. However, web pages contain information which is not main content, for instance, online advertising, related information, website menu, navigation links, and so on. This kind of information has become the noise of creating Web search engine and decreases the accuracy of search result. Therefore, extraction of the main contents is crucial to improve the quality of blog search. A wrapper is a program that selectively extracts a necessary part (component) from Web pages. By ranking the importance of the components, we propose a component-based search engine in which the content components gain a high score in the search results. Thus, the required segments for a query can be obtained without using a wrapper.

The authors of the present paper have been working on a new search model [1], [2]. Tourism information was chosen to explain the effectiveness of the model. The blog site “Kyushu seifuku Blog” was chosen as analysis target of this research. This blog site is about the tourism information on the Kyushu area, which is published by Kyushu Tourism Promotion Organization¹. Every blog entity on the site is composed of simple introduction and link. The details of the blog entities are open to the public on other external sites which are independent to the Kyushu Tourism Promotion Organization. We collected 1,303 blog entities and saved as html files and extracted 136,368 components from these blog.

II. RELATED WORK

A. Keyword Search for Semi-Structured Documents

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 identifiers.

¹<http://www.welcomekyushu.jp/>

The first use was in the SMART Information Retrieval System [3]. According to the increase of HTML documents and XML documents, search engine for semi-structured documents are gaining attention. Moreover, it has been a big issue to integrate structure search and keyword search. For example, Seki et.al. [4] realised a faceted search for sub-structures using multiple vector space model search engines. It has become very popular to utilize vector space model to extract contents from semi-structured documents. Chidlovskii [5] described a method about information extraction from HTML/XML trees. Hu and his colleagues tried to formulate the semi-automatic construction parts as an algorithm [6]. Cooper and his colleagues proposed an indexing structure in order to achieve high query performance using semi-structured data repositories [7]. Moreover, Zhang, Ohmori and Hoshi analyzed the efficiency of keyword-based search for Hybrid XML-relational databases [8]. In order to improve the retrieval efficiency, Hatano and his colleagues proposed a method for determining fine grained results for keyword-based XML document retrieval [9].

All of these papers demonstrate how to integrate efficiently semi-structured document search into keyword document search. However, most of these approaches consider fixed XML formats which cannot deal with a variety of structures often seen in various blog sites. Each blog site provides completely different structures. Sometimes, bloggers can settle their own favorite format. So, we need structure-independent methods for blog search.

B. Blog Spams and Content Extraction

Blog sites are written in HTML. They are typical semi-structured documents. However, blog sites are not official sites. Official sites have documents with specific and clear purpose. Blogs are mixed by different purpose information such as the comments of blogger and company online advertising. Therefore it is difficult to evaluate the importance of a blog page as a single entity. We focus on the smallest plain text areas, which we call components, by which the pager is evaluated as a single entity.

One of the main difficulties in dealing with blogs is the increase of spams. There are a lot of researches about detection and the removal of the blog spam to improve the quality of search [10]. Even if we would succeed in choosing non-spam blogs, there remains a difficulty in extracting the real contents from the blog pages. Indeed, the target 1303 web pages, the present paper is analyzing, are chosen by experts of the tourism

organization. They are not spam at all. However they still contain many navigational contents and advertisements in the pages. So, we will lose the feature of the page if we use all the words that appear in the page as the index words. To separate the real contents and other is a big problem we have to face. There are a few researches in extracting the pure contents using shallow features [11], [12].

III. COMPONENT INDEX

In order to create a blog search engine, it is necessary to extract the useful and valuable information from that blog. A large number of blog entries are written by the blog authors, and these constitute the main contents of the blog, and represent useful and valuable information. However, the main contents are often accompanied by a large amount of noise such as online advertisements, links to other blogs, copyright notices or the comments of readers. These noise data seriously harm the extraction of the blog's primary content. The top-ranked search results might not be pertinent for the query due to these noise. This is a big problem we must overcome to realise an ideal blog search engine. If all blogs were written in the same uniform template, it would be very easy separate contents and noise. However, there are a large number of blog sites, which have different templates. Moreover, blog authors can set up personalized templates according to their individual preferences. If we cannot distinguish the main contents and other noise, all the words in the HTML page will be recognized and indexed as feature words by which search is performed. In the end, the top ranked search results might be irrelevant.

The present paper considers a different approach to realise a high quality blog search. We believe that we can distinguish enhanced blogs and thin blogs even if we do not have a complete extraction program for article contents.

In the present paper, a web page is caught as a set in a lot of small text parts, which contain no tags. In other words, we focus on the leaves of HTML-trees. We do not try to separate the contents and other parts. Instead, we introduce a score for all these components. According to this score, the weight of words in the article become high and those in other parts become low. As the result, the rich content blogs appear at the top ranking in the search result.

Fig. 1 is an article in a tourism blog. It contains the navigational links of the site ①, links to the archives ②, links to the latest contents ③, the title of the article ④, and the main contents of the article ⑤. Fig. 2 shows the web page as an html-tree, where the square boxes highlight the main contents, where square boxes represents those content text areas. We call these small text areas as "components" and realised a search engine for these components by indexing them. We used GETA system² to realise the search engine. Fig. 3 displays the index file of the search engine. The lines starting with "@" symbol show the indexed objects, i.e. components. For example, "@1-1" and "@1-2" represents the first and the second text area of the first HTML file (Fig. 1). Other lines display index keywords

and their frequencies in the text area. The keywords with "h:" represents the id of the HTML file. The keywords with "p:" represents the X-path by which the component can be reached from the root of the HTML-tree structure. For example, the keyword "p:/html[2]/body[4]/div[1]/div[1]/h3[1]/" in the second component "@1-2" represents the X-path to the text area ④, in Fig. 1, which is the headline of the article "Huis Ten Bosh is now blooming with roses!".



Fig. 1. A Tourism Blog

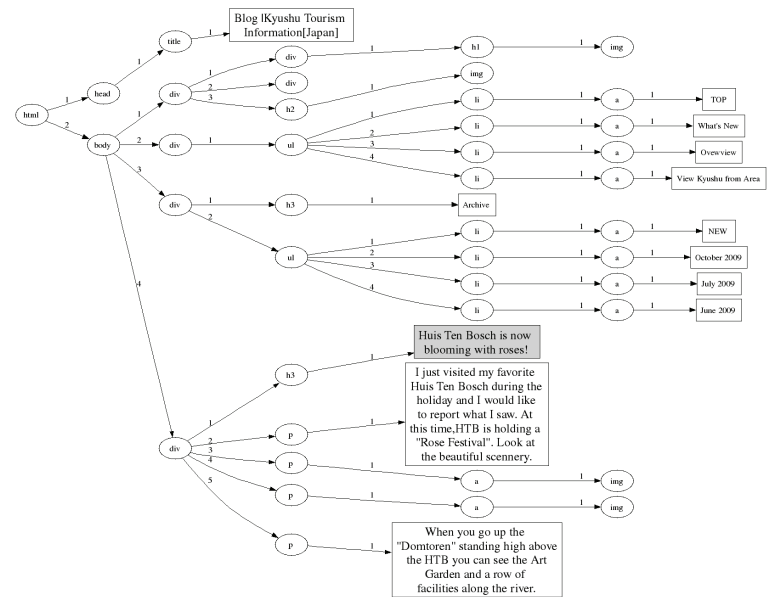


Fig. 2. HTML Tree

The anchor text that appears under the anchor tag is excluded from the components. Because the linked text is usually relevant to the other web pages, it is not useful contents to the search engine. Actually, most of them were displayed in the top, bottom, left and right of a blog page, what are links to top page, other blogs, or links to the online advertisement. Some anchor texts might correspond to internal link within the page. However, these anchor text usually contain only one or two words. Moreover, these words appear in the text area which is the target of the link. Therefore, eliminating these

²<http://geta.ex.nii.ac.jp>

```

@1-1
1 h:1
1 p:/html[1]/head[1]/title[1]
1 Blog
1 Kyushu
1 Tourism
1 ...
@1-2
1 h:1
1 p:/html[2]/body[4]/div[1]/div[1]/h3[1]/
1 Houis
1 Ten
2 Bosh
2 is
1 ...
@1-3
1 h:1
1 p:/html[2]/body[4]/div[1]/div[2]/p[1]/
3 I
1 just
1 visit
1 ...
:
:

```

Fig. 3. Component Index

internal link anchor texts does not loose any information from the page.

IV. COMPONENT SCORES

Lots of calculation methods were proposed to evaluate how important a word is to a document in a collection, such as “tf*idf” and the calculation method of SMART system [3], which are suitable long document. Comparing with the generic document, the sizes of the most of the components in this paper are a lot smaller. Therefore, “tf*idf” is not suitable these components.

We introduce the following formula for calculating the score of the components.

$$score(C_i) = NW(C_i) * depth(C_i),$$

where $NW(C_i)$ is the number of the distinct words in the component C_i and the depth $depth(C_i)$ is the length of the path to HTML tree’s root. We found that the individual and detailed content is always put on the deeper nodes of the HTML tree, therefore, we use the depth to calculate the SC. On the other hand, the noise such as online advertisements is always put on the shallow nodes, which have common templates. Moreover, individual entities are written in original contents, which have rich vocabulary. The score of a web page is defined as the total scores of all the components in the page.

V. SEARCH RESULTS FOR “NOODLE”

We compared the SMART base standard score $S(D)$ and the component score $C(D)$ of html files D . Considering

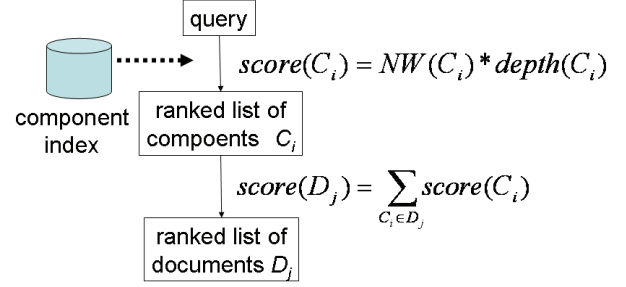


Fig. 4. Score of Components and Documents

practical situation, we conducted empirical evaluation using the following 22 keywords which are often seen in blog articles. These keywords are related to food, restaurant and gourmet.

noodle, soup, “ramen”, dish, lunch, “chaashuu”, vegetable, meat, pork, onion, “sake”, “udon”, gourmet, chicken, fish, “soy sauce”, egg, restaurant, meal, beef, rice, “tonkotsu”

For each 22 keyword w_i , firstly, we retrieved all the blog articles $D_{i,1}, \dots, D_{i,N_i}$ that contain the keyword w_i . Secondly, we calculated the scores S-score(D_i) and C-score(D_i) together with their ranking S-rank(D_i) and C-rank(D_i). Then, we manually checked each pages $D_{i,j}$ when they have a large difference between S-rank(D_i) and C-rank(D_i). According to the limit of space, we explain the analysis for the word “food” and 121 articles that contains the word.

Fig. 5 displays the scores of the 121 pages, where the x-axis represents the standard ranking (S-rank) and the y-axis represents the component-based ranking (C-rank) of each page. Most pages locate on the diagonal line and have almost similar rankings in S-rank and C-rank. However, there are some pages whose rankings are greatly changed. We selected top 12 pages among those pages and marked by square points. Table II displays the original data of those pages.

In the page where the ranking falls down, C-score is small. On the other hand, C-score is large for the page where the order went up. TableII displays typical components of each page which contain the query word (noodle). We see that the page where the ranking falls down (rank-descending page) contains a short sentence. We see that the page where the ranking went up is an article which contains given keyword and is written by a detailed content. It is worth-while to note that the rank-descending pages contain the query word, as shown in TableII, and that their C-score are small. This implies that short sentences shown as the snippets in TableII are the reason why those pages matches the query, as long as the components are concerned. The reason why those rank-descending pages gained high ranking in the standard ranking is not that their main contents contain the keyword but that they contain the keyword in navigation area or in left or right frames. On the other hand, the rank-ascending pages gained the high ranking because they are written with detailed contents as we can see Table II.

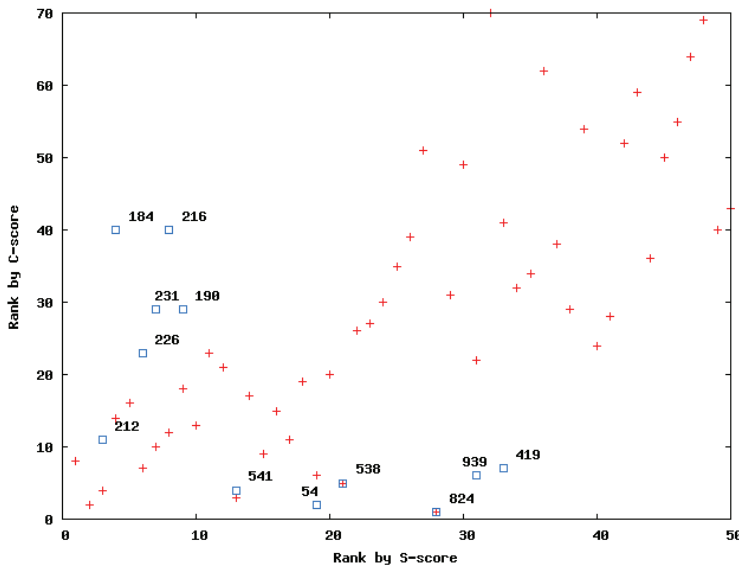


Fig. 5. Comparison of S-rank and C-rank

TABLE I
SCORE AND RANKING

HTML	S-rank	C-rank	Rank Diff	S-score	C-score
184.html	4	40	-36	0.0406	153
216.html	8	40	-32	0.0367	153
231.html	7	29	-22	0.0374	207
190.html	9	29	-20	0.0366	207
226.html	6	23	-17	0.0389	252
212.html	3	11	-8	0.0420	476
541.html	13	4	9	0.0348	880
538.html	21	5	16	0.0295	868
54.html	19	2	17	0.0309	1125
939.html	31	6	25	0.0260	830
419.html	33	7	26	0.0252	793
824.html	28	1	27	0.0272	1249

We displayed only one component in Table II. But the system already has all the components of the page and the scores of those components. So, we can display other appropriate components as snippets, which helps users to grasp the contents of the page before they actually read.

Top 6 pages (184.html, 216.html, 231.html, 190.html, 226.html, 212.html) that gained a big loss in the ranking, are in a private blog ³. Their main articles are very short. However, they use a large area in the page for the pictures and for the links to related articles and to advertisement of affiliate (See Fig. 6).

A rank-ascending page, for example 419.html ⁴, contains a long sentence with the keyword and most of other contents concern personal experience and reproof of the blogger (See Fig. 7).

According to this case study, we observe that the pages which contain enhanced contents gained high ranks in the component-based ranking, while the pages whose related information is more than the article itself becomes a subordinate

TABLE II
DESCENDING PAGES(-) AND ASCENDING PAGES(+)

inc/ dec	comp id	content
-	184-25	太麺の丸麺。
-	216-27	前回の麺三味は
-	231-38	前回の麺三味は。
-	190-25	ただ薄いかな？と思いながら麺へ
-	226-22	前回の麺三味は
-	212-81	モツ＆ちゃんぽん麺なんて最高ですね～（●^o^●）
+	541-11	私はかなり久しぶりに来たんですが、ランチメニューを見ると、日替わりのセット（800 円）に担々麺（700 円）、酢豚や麻婆豆腐（各 950 円）といった具合に、なかなか幅広い選択肢があるようです♪
+	538-27	チャーシューも柔らかく旨みがあって美味しんですが、この麺と野菜のボリュームからするとやや寂しいので、トッピングの「細切れ豚」（100 円）を頼むのが正解かもしれませんね
+	54-63	とり天も冷麺も大好きな私！こりや参差せねばいかにです！
+	939-119	野菜、魚介の旨みが凝縮されてミルクのようにコクのあるクリーミーなスープは最高。スープに馴染んだ太麺の食感も最高。
+	419-27	坦々麺に、たっぷりのカボチャピューレと、素揚げしたカボチャが入っています♪
+	824-25	すると、麺に絡まるんです



Fig. 6. Page with Navigational Contents(184.html)

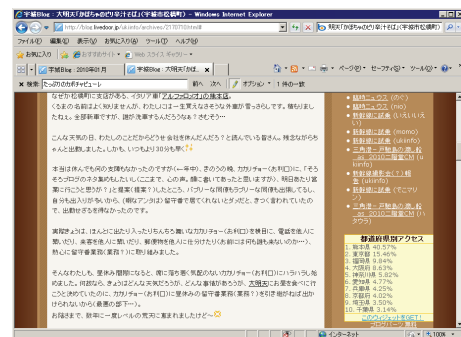


Fig. 7. Page with Rich Contents(419.html)

³<http://bakudankozo.blog118.fc2.com/>

⁴<http://blog.livedoor.jp/ukiinfo/archives/2170710.html>

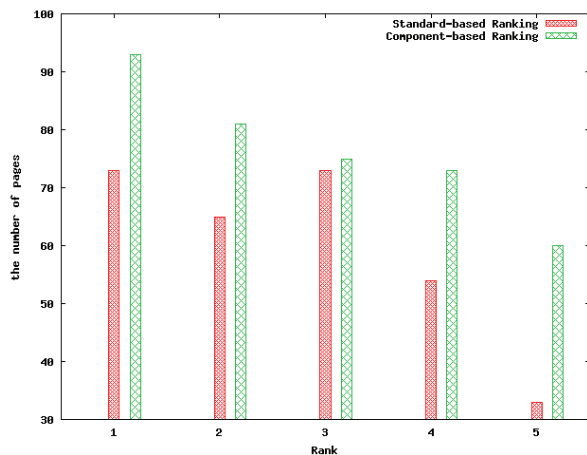


Fig. 8. User Preference of Standard Ranking and Component-based Ranking

position. This case study can be a clue for the effectiveness of the proposed ranking for detecting rich content blogs. We observed the similar pattern for other 21 cases of the keywords.

VI. QUANTITATIVE EVALUATION OF RANKING

We realised a standard search engine and a component-based search engine for 1303 blogs using GETA. We performed a quantitative evaluation of rankings using the 22 keywords as queries.

We compare the ranking of the pages in S-rank and C-rank. Actually, we constructed an experiment system for testee to select appropriate pages among

- (A) top 5 pages in S-rank and
- (B) top 5 pages in C-rank.

These pages are shuffled random and displayed as a search result for each query word for 22 keywords. We hide the rank of each search result. The testees do not know which page is chosen by S-rank or by C-rank. We displayed the components of the highest score as snippets in (A) and (B). Testees read the snippets and view the page, if they are interested, to choose the pages according to their preference. Fig. 8 shows the result. Component-based ranking obtained a better preference by testees in all ranks.

The total number of articles is not so large. But, those blog articles are collected from 144 sites and written different format. So, we think that this diversity supports our observation of the effectiveness of the proposed method.

VII. CONCLUSION AND FURTHER WORK

Many people are using blogs as a means for publishing their opinions and recognize them as useful informal information sources to be searched for. However, a blog page contains various information besides the article itself. It is difficult to extract only the contents and utilize these contents, since there is no general rule to describe blog pages. In fact, blog pages have different format even if they are written in the same site. The present paper introduced a novel scoring method for a blog page focused on their components. This score and

the ranking by this score succeeds to distinguish pages with enhanced articles from thin pages. Empirical evaluation are shown by constructing a search engine for tourism blogs. Effectiveness of the proposed method is confirmed through a detailed comparison of rankings of search result as well as quantitative evaluation.

We considered the leaves of an HTML-tree as components in the present paper. We excluded the anchor texts from components, since they do not contain the actual content of the page, but are links to external information. There may be other possibilities to formalize components. Inline tags, such as BR tag, and punctuation marks will be useful to characterize the real article area. Such shallow feature will improve the score as indicated in [12].

The authors appreciate the valuable comments from reviewers.

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] G. Salton, The SMART Retrieval System—Experiments in Automatic Document Processing, Prentice-Hall, Inc. (1971)
- [4] Takahiro Seki, Taiki Wada, Yasuhiro Yamada, Nozomi Ytow and Sachio Hirokawa, Multiple Viewed Search Engine for e-Journal — a Case Study on Zoological Science, Proc. of the 12th International Conference on Human-Computer Interaction, Lecture Notes in Computer Science, Vol. 4553/2007, pp. 989-998, July 2007.
- [5] Boris Chidlovskii: Information Extraction from Tree Documents by Learning Subtree Delimiters. IIWeb 2003: 3-8
- [6] Jin Hu, Hedenari Kiyomitsu, Kazuhiro Ohtsuki and Junya Morishita, Operations for Retrieving a Potion from Semi-structured(in Japanese) Resources,3C-i12, DEWS2006, 2006
- [7] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason and M. Shadmon," A fast index for semistructured data, " Proceedings of the 27th VLDB Conference, 2001.
- [8] Liru Zhang, Tadashi Ohmori and Mamoru Hoshi, Keyword-based Search for Hybrid XML-Relational Databases (in Japanese), DEWS2008 C6-2, 2008
- [9] Kenji Hatano, Hiroko Kinutani, Masatoshi Yoshikawa and Shunsuke Uemura, Determining Fine-grained Results for Keyword-based XML Document Retrieval (in Japanese), DBSJ Letters Vol.2, No.1, 2003
- [10] <http://airweb.cse.lehigh.edu/2008/>
- [11] <http://search.cpan.org/dist/HTML-ExtractContent/>
- [12] Christian Kohlschutter, Peter Fankhauser, Wolfgang Nejdl, Boilerplate detection using shallow text features, Proc. WSDM'2010, pp.441-450, 2010