DOI : 10.3745/JIPS.2009.5.3.159

Journal of Information Processing Systems, Vol.5, No.3, September 2009 **159**

Automatic In-Text Keyword Tagging based on Information Retrieval

Jinsuk Kim*, Du-Seok Jin*, KwangYoung Kim* and Ho-Seop Choe*

Abstract: As shown in Wikipedia, tagging or cross-linking through major keywords in a document collection improves not only the readability of documents but also responsive and adaptive navigation among related documents. In recent years, the Semantic Web has increased the importance of social tagging as a key feature of the Web 2.0 and, as its crucial phenotype, Tag Cloud has emerged to the public. In this paper we provide an efficient method of automated in-text keyword tagging based on large-scale controlled term collection or keyword dictionary, where the computational complexity of O(mN) – if a pattern matching algorithm is used – can be reduced to O(mlogN) – if an Information Retrieval technique is adopted – while m is the length of target document and N is the total number of candidate terms to be tagged. The result shows that automatic in-text tagging with keywords filtered by Information Retrieval speeds up to about 6 ~ 40 times compared with the fastest pattern matching algorithm.

Keywords: *Automatic In-Text Keyword Tagging, Information Retrieval, Pattern Matching, Boyer-Moore-Horspool Algorithm, Keyword Dictionary, Cross-Referencing, in-text content link*

1. Introduction

Since the World Wide Web has emerged, tagging or linking across the HTML texts has played a crucial role in the success of the Internet. In the late 1990s, Google noticed the importance of such tags (in-text content hyper-links) in HTML texts and made a successful improvement in web page search algorithms, in the form of PageRank [1], resulting in a paragon in the Internet business. Furthermore, Web2.0 has increased *tags* or *tagging* as one of the biggest issues in the Internet society. Tagging as a phenolmenon corresponds with a Web 2.0 mentality that users can create not only contents but a richer, more adaptive and responsive way to navigate and search both existing and new media. However, the problem is that most tagging activities remain manual due to immature techniques in the natural language processing arena. For example, Wikipedia[‡] is regarded as one of the most successful Web 2.0 services, but tagging or cross-linking related items with each other in Wiki articles is still created manually.

In this paper, we present an efficient method of automated in-text keyword tagging which can replace or supplement manual tagging processes (in the scope of this paper, in-text tagging can be regarded as automated creation of keyword-level links among articles in encyclopedia or in a bunch of documents such as Wikipedia). Automatic in-text keyword tagging is important since the cost of manual in-text keyword tagging is very expensive. For example, the Encyclopedia of Korean Local Culture[§] is a compilation of XML documents of which major keywords such as people and location are fully tagged manually [2]. In this project, all proper nouns are in-text tagged to enhance hypertext-based cross-referencing. However, the cost of tagging a document is higher than that of writing itself (*personal communication*). In this case, for example, person name entries of proper nouns related to the city can be easily collected and constructed to a proper noun dictionary. Using this vocabulary, it is expected that an automated in-text keyword tagging process can be applied. And then by supplementing the manual tagging with automated in-text keyword tagging, the cost and time of tagging can be remarkably reduced. Furthermore by reducing human born errors, tagging precision is expected to be improved, too.

Another example is a bibliography database of scientific journal articles. Usually, each bibliographic data contains a KEYWORDS section usually assigned by the authors, and by using this field, the bibliographic information can be improved with flourished hypertext links for keywords in abstract or even in the full text fields. Introducing a dictionary of scientific terms may also enrich the readability and responsive navigation of the journal articles.

This study presents experimental results of an automated in-text keyword tagging technique with a large-scale key-

[‡] http://www.wikipedia.org

Manuscript received June 16, 2009; revised August 6, 2009; accepted August 24, 2009.

Corresponding Author: Ho-Seop Choe

* Department of Information Technology Research, Knowledge Information Center, Korea Institute of Science & Technology Information (KISTI), Gwahangno 335, Yuseong-gu, Daejeon, Republic of Korea ({jinsuk, dsjin, kykim, hschoe}@kisti.re.kr)

[§] http://www.grandculture.net

word dictionary. Then we analyze some analyses of the complexity reduction based on experimental results. The feasibility of our in-text tagging method implies that automated in-text keyword tagging can be applied to various areas such as XML compilation systems, context-based advertising, automated in-text keyword links on the fly, automatic keyword extraction based on controlled vocabulary, cross-referencing in encyclopedia [3], and HTML documents transformed automatically from plain texts [7-10].

2. Related Work

There are two kinds of keyword tagging approaches depending on location of the tagged keywords: ***out-of-text*** keyword tagging and ***in-text*** keyword tagging. Out-of-text keyword tagging approaches maintain tagged keywords externally to the text as shown in cases such as book index, author-assigned keywords in scientific journal articles, and Web2.0 tags. On the contrary, in-text keyword tagging marks up keywords in the text directly as shown in HTML texts. Among information sharing frameworks, for another example, FTP and GOPHER are protocols based on out-of-text keyword tags, while HTTP allows document navigation through in-text keyword tags.

2.1 Out-of-text Keyword Tagging

Out-of-text tags refer to metadata about a document which are placed out of the text body. Bibliographic databases, author keywords in journal articles, and indexes of books are easily noticeable examples of out-of-text tags. In the advent of Web2.0, tagging systems allow users to annotate digital resources with tags (keywords) and share their annotations with other users [11]. This kind of tagging is referred to by several names such as *collaborative tagging*, *social classification*, *social indexing*, and *folksonomy*. This is the most well-known example of out-of-text keyword tagging and currently draws deep interest in research communities.

Out-of-text keyword tags are usually maintained manually in practical tagging systems [11]. To resolve the problem, recent research interests include automatic tagging for blog posts [12,13], real-time recommendation in the tag space [14,15, 18], automating the process of tag assignment and automated annotation [16,17], *etc*. Other research issues on tagging services include improving the quality of searching [18] and analyzing the usage patterns of tagging systems [19].

2.2 In-text Keyword Tagging

In-text tags are keywords that are tagged directly in the text body. They are expressed as hypertext links in the HTML texts. As the World Wide Web grows exponentially, the hypertext links connect heterogeneous digital resources with HTML's inherent feature of in-text content links. Since these links are manually generated in-context text links, web search engines have noticed their importance and have exploited them as one of the major features of the ranking algorithms. Though in-text content links in HTML texts are manually generated, there has been research of off-line techniques to automatically transform mass plain texts to HTML documents with cross-referencing links [7-10].

The *in-text advertising*, which is one of the on-line in-text keyword tagging services, has also been noticed for its importance by advertising companies such as Kontera[**],

Brown Argus Butterfly Sees Positive Effects Of Climate Change

Global warming is generally thought to have a negative affect on the habitats of many animals and plants. Not for the Brown Argus butterfly, however. This insect seems to be bucking the trend and expanding its numbers quicker and more effectively, according to new research.

The Brown Argus butterfly Aricia agestis has expanded northwards in Britain during the last 30 years. It is thought that the recent expansion of the species is due to the increasing summer temperatures caused by global warming.

(a) Palin Text

Brown Argus Butterfly Sees Positive Effects Of Climate Change

Global warming is generally thought to have a negative affect on the habitats of many animals and plants. Not for the **Brown Argus butterfly**, however. This insect seems to be bucking the trend and expanding its numbers quicker and more effectively, according to new research.

The **Brown Argus butterfly** ***Aricia agestis*** has expanded northwards in Britain during the last 30 years. It is thought that the recent expansion of the species is due to the increasing summer temperatures caused by **global warming**.

(b) Text with in-text keyword tagging

Fig. 1. An example of improved readability of a document by tagging major keywords in-text

[**] http://www.kontera.com

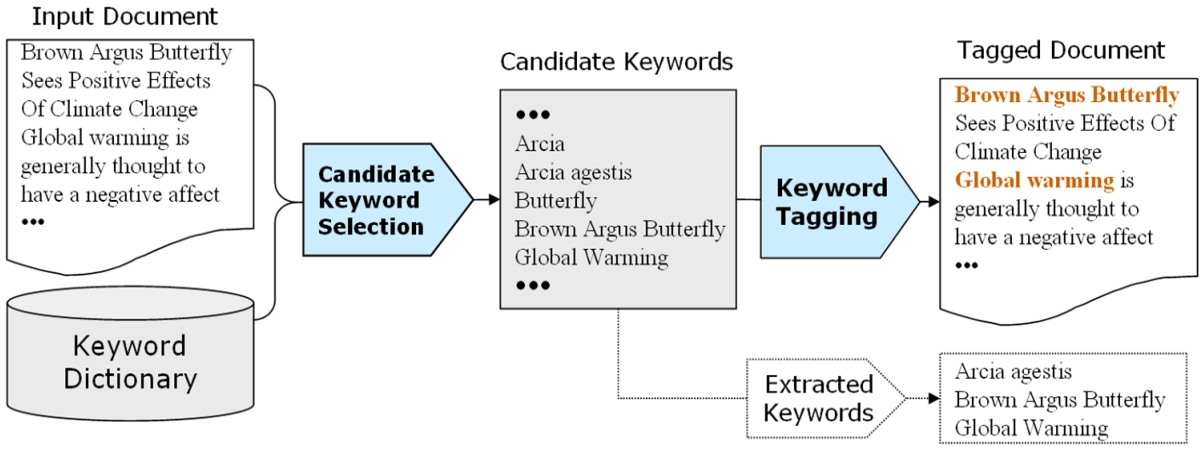


Fig. 2. Two steps of Automated In-Text Keyword Tagging: (1) **selection of candidate keywords**; and (2) **keyword tagging** (and/or extraction)

LinkAdAge[††], LinkXL[‡‡], and Vibrant[§§]. In-text advertising is a form of contextual advertising where links to sponsor sites appear in the middle of normal web pages on a website as part of the content by linking to specific words within the text. It analyzes given HTML text, selects semantically appropriate lists of keywords or phrases extracted from the HTML text, and assigns link(s) to the sponsor sites. In most in-text advertising, the text associated with an advertisement is identified by a double-underline to differentiate it from regular hyperlinks, and clicking it or positioning the mouse cursor over it pops-up an in-page window containing advertising content.

Although out-of-text keyword tagging is a major issue in Web2.0, there have been little research on the efficacy and/or efficiency of automatic in-text keyword tagging algorithms. In this paper, we will present an efficient method of on-line in-text keyword tagging with a large-scale keyword dictionary using information retrieval.

3. Automatic In-Text Keyword Tagging

Tags can serve as informal metadata for objects such as web pages and multimedia data. If a method to automatically generate tagged keywords in a text can be given, it is possible to provide more adaptive and responsive ways to navigate among related documents. **Figure 1** shows an example of improved readability of a document by tagging the major keywords *in-text*. Just glancing the tagged text in **Figure 1 (b)**, readers may be able to catch that the subject of the document is, "***Global warming*** and its effect on ***Brown Argus butterfly*** (Aricia agestis)". Furthermore, in-context cross-referencing in **Figure 1 (b)** will be able to aid users to navigate more related documents in the text collection.

This study presents an efficient method of automatic in-text keyword tagging which consists of two steps; (1) **selection of candidate keywords** and (2) **keyword tagging** using candidates from the first step (*see* **Figure 2**).

As shown in **Figure 2** (*see also* **Algorithms 1&2** at the end of this section), the first step selects candidate keywords from the keyword dictionary by comparing the input document and *whole* terms in the dictionary. The second step extracts exact keywords and tags those in the input text. In the first step, if a string matching algorithm is used, selection of candidates from the dictionary is proportional to the length (m) of the input document and the number (N) of keyword entries in the keyword dictionary, resulting in the computational complexity of $O(mN)$. At the second stage, tagging with given keywords is proportional to the document length, m, hence the complexity is $O(m)$. Since, m is to be much smaller than N, overall complexity of automated keyword tagging is overwhelmed by the first step, hence the overall complexity of $O(mN)$.

As this brief analysis of the computational complexity shows, the bottleneck of automated keyword tagging is the first step, where a set of candidate keywords is constructed by comparing input text and terms in the keyword dictionary. To evaluate possible solutions for this problem, we tested three methods; (A) a general pattern matching algorithm by applying string::find() function[***] in C++'s standard string class, (B) an advanced string matching method by

[††] http://www.linkadage.com
[‡‡] http://www.linkxl.com
[§§] http://www.vibrantmedia.co.uk

[***] In addition to C++'s string::find(), string matching functions from other programming languages such as C language's strstr() function and JAVA's String:: matches() was tested. C strstr() is slightly faster than C++'s string::find() and JAVA's String:: matches() is two to three times slower than C++'s string:: find() function. Due to implementation issue, we used C++'s string::find() in this experiment.

implementing Boyer-Moore-Horspool (BMH) algorithm [4,5], and (C) filtering whole terms to build a set of candidate keywords using inverted index of an information retrieval system. In methods A and B, as mentioned above, the time complexity for comparing the input document and keyword dictionary is O(mN) since the basic pattern matching algorithm has O(m) complexity and it should be executed N times (**Algorithm 1**). In method **C**, an open-source Information Retrieval & Management System KRISTAL-IRMS [6] was used to store and index whole terms in the keyword dictionary, and to retrieve candidate terms using a vector space model by querying the whole text of the input document as a query (**Algorithm 2**).

In method C, the inverted index contains, in sorted order, all the unique strings which appear in the text collection (all terms in the keyword dictionary in this study), together with a pointer to posting list. Since the index is accessed by a binary 'divide and conquer' search algorithm, the time complexity of retrieval for each keyword is O($k$log$L$) where k is the average number of unique tokens in each document and L is the total number of unique tokens in the text collection. In this study, k is a small constant since each document is a scientific term which consists of only a few words and $k << \log L$, the time complexity of building a set of candidate keywords using information retrieval is O(log$L$). And then, since L is proportional to the total number of scientific terms, N, we can conclude that O(log$L$) = O (log$N$). Hence, the time complexity of the first step in method C is O(log N) where N is the size of the database *i.e.* the number of documents in the database [5]. Therefore, given a document with length m, the overall order of time complexity in method C is O($m$log$N$). The experimental result supports this kind of significant reduction in computational complexity.

Algorithm 1. Automated In-Text Keyword Tagging with Pattern Matching

On-line computation:

[Step 1: *selection of candidate keywords*]
1: Given a collection of keywords, $K = \{k_1, k_2, k_3, \ldots, k_N\}$
2: For each keyword of K,
Extract all tokens, $Q=\{q_1, q_2, q_3, \ldots, q_r\}$
3a: **Add** to candidate keyword set C, if all q exists in the input text t by checking with C++ string::find()
3b: **Add** to candidate keyword set C, if all q exists in the input text t by checking with Boyer-Moore-Horspool algorithm

[Step 2: *keyword tagging*]
4: **Sort** C according to descending order of byte lengths of keywords.
5: For each keyword of C, **find** all locations and **make** link(s) in the input text t.

Algorithm 2. Automated In-Text Keyword Tagging with Information Retrieval(IR)

Off-line computation:

1: Given a collection of keywords, $D = \{k_1, k_2, k_3, \ldots, k_N\}$
2: Apply stemming on tokens of all keywords.
3: **Build** inverted index with KRISTAL; $I = \{<t_1, K_1>, <t_2, K_2>, <t_3, K_3>, \ldots, <t_m, K_m>\}$ where K_i is set of keywords that include term t_i.

On-line computation:

[Step 1: *selection of candidate keywords*]
1: Given an input text t,
2: **Extract** all tokens and apply stemming on them; $T=\{w_1, w_2, w_3, \ldots, w_n\}$
3: **Select** top-r high frequency tokens from T; $Q=\{q_1, q_2, q_3, \ldots, q_r\}$
4: With vector space model, **retrieve** top-k keywords with the highest similarities;
$C = Q \cap I$ where $k \leq 2{,}000$.

[Step 2: *keyword tagging*]
5: Sort C according to descending order of byte lengths of keywords.
6: For each keyword of C, **find** all locations and **make** link(s) in the input text t.

4. Experiments

4.1 Experimental Environment

All experiments were conducted on a PC with a 2.6GHz Pentium IV CPU, 1GB of RAM, 80GB SATA (Serial ATA) HDD and Linux operating system. Test programs were written with C++ language and compiled with GNU g++ compiler. All the tests were carried out while the machine was under light load.

4.2 Test Data

Input documents were prepared from scientific journal articles according to their lengths. We extracted text parts of PDF files and made the test documents, the lengths of which were 1kb, 5kb, 10kb, 50kb, and 100kb, respectively. The content of each input text is written in English. These documents were used to evaluate the effect of document

length on automatic in-text keyword tagging.

4.3 Keyword Dictionary

A Keyword dictionary was constructed by extracting 400 thousand terms from a dictionary of science and technology. This dictionary has been built by KISTI and consists of about 400 thousand English terms and their corresponding Korean terms. We randomly extracted 10,000, 50,000, 100,000, 200,000, and 400,000 English terms and used them to evaluate the effect of dictionary size in the automated in-text keyword tagging.

4.4 Selection of Candidate Keywords

• **Method A**. **Simple String Matching** between input document and whole terms in the keyword dictionary

The string class of C++ programming language provides a substring search function, string::find(). In Method A, each term in the keyword dictionary was tokenized by blanks and checked whether it occurs in the input document using string::find() function. If all tokens occur in the text, the term is regarded as a candidate keyword (line 3a of **Algorithm 1**). The keyword dictionary was stored in a text file with each term in a line. The time to read this file was not included in the experimental results.

• **Method B**. **Advanced String Matching** between input document and whole terms in the keyword dictionary

This method is the same as Method A except that the Boyer-Moore-Horspool (BMH) string matching algorithm is applied, instead of string::find() to compare input text and terms in the keyword dictionary (line 3b of **Algorithm 1**). The BMH algorithm was implemented with C++ language slightly modified from the source code given in [5].

• **Method C**. **Information Retrieval** to filter whole terms in the keyword dictionary to build a set of candidate terms by querying the input document as query

In this method, all terms from the keyword dictionary are stored into and indexed by an open-source information retrieval engine, KRISTAL-IRMS [6]. Upon tagging request, the input document is used to query the dictionary database and to retrieve top-ranked documents (keywords in this case) using the vector space model supplied in the KRISTAL engine. These retrieved documents are regarded as the candidate keywords. In this paper, we restricted the number of candidate keywords to 2,000 top-ranked terms returned by the KRISTAL system (step 1 of **Algorithm 2**). The Memory database feature provided by KRISTAL was also used to speed up fetching result terms from the dictionary database.

5. Experimental Results

5.1 Input Document Length (*m*) vs. Automated In-Text Keyword Tagging

The first experiment is to measure the effect of the text length (m) of the input document on the efficiency of in-text keyword tagging. In **Figure 3**, results are shown according to three methods of building the set of candidate keywords. In this experiment, the number of terms in the keyword dictionary was fixed to 50,000 entries and the length of input text was varied from 500 bytes to 200 kilo bytes.

Method B with BMH algorithm used in filtering candidate keywords from whole term collection outperforms Method A with C++ string::find() function by 2.8 times on average. Method C with IR filtering in candidate keywords outperforms Method A and Method B by 13.8 and 5.0 times on average, respectively. For all three methods, tagging time was approximately linearly proportional to the text size (m) as shown in **Figure 3**, meaning that the time complexity of automated in-text keyword tagging is proportional to O(m).

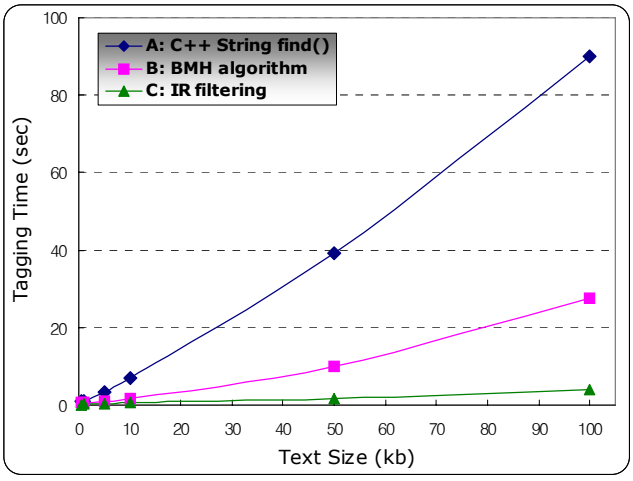


Fig. 3. Text size vs. in-text tagging time (dictionary size = 50,000 terms)

5.2 Number of Dictionary Items (*N*) vs. Automated In-Text Keyword Tagging

This experiment is to measure the effect of the keyword dictionary size (N) on the efficiency of automated in-text keyword tagging. In **Figure 4**, results are shown according

to three methods of building the set of candidate keywords. In this experiment, the text length of the input document was fixed to 5 kilo bytes and the size of the keyword dictionary was varied from 10,000 to 400,000 entries of scientific terms.

Method B (BMH algorithm) outperforms Method A (string::find()) by 3.2 times on average. Method C (IR filtering) outperforms Method A and Method B by 17.8 and 5.6 times on average, respectively. For string matching algorithms (Methods A and B), tagging time is approximately linearly proportional to the dictionary size (N) as shown in **Figure 4**. On the contrary, compared with Methods A and B, Method C shows nearly constant tagging time for all dictionary sizes. At the dictionary size of 50,000 entries, Method B outperforms Method A by 2.7 times, but at 400,000 entries, Method C is more than 40 times faster than Method A.

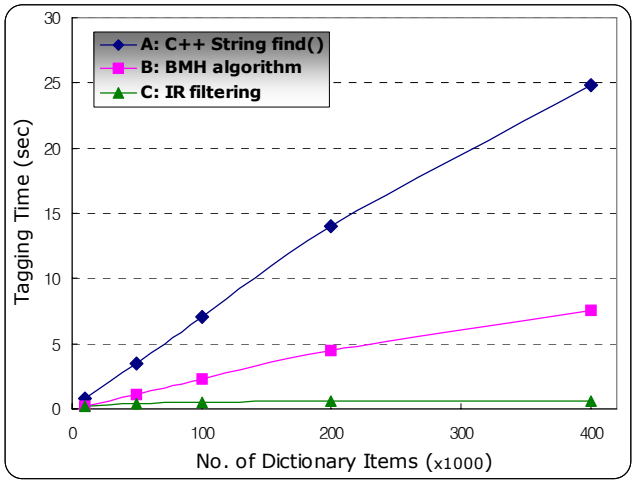


Fig. 4. Dictionary size vs. in-text tagging time (input text length = 5kb)

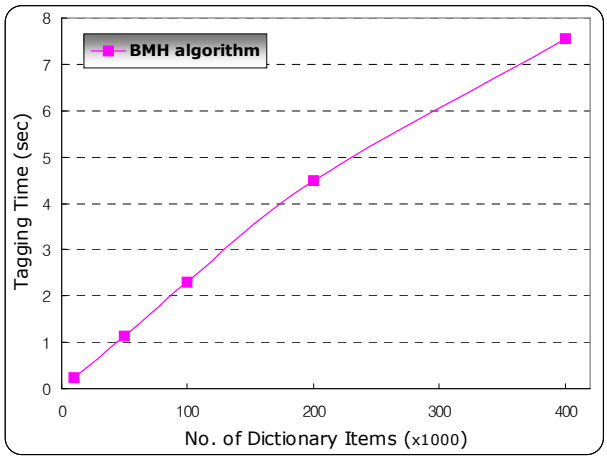
Figure 5 compares a more detailed view of the effect of dictionary size on automatic in-text keyword tagging for Methods B and C. For BMH algorithm (**Figure 5(a)**), tagging time is linearly proportional to dictionary size. On the contrary, Method C shows logarithmic graph between tagging time and dictionary size (**Figure 5(b)**). This result supports our analysis that string matching based candidate keyword selection has time complexity of O(N) and IR-based filtering's time complexity is reduced to O(log$N$).

6. Conclusions and Future Directions

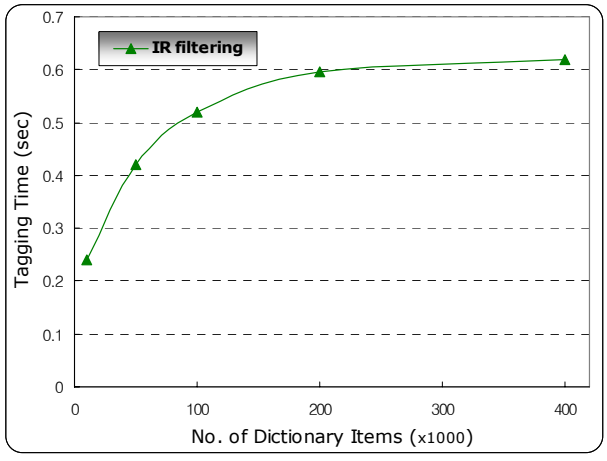
Figures 3, 4, and **5** show that the time complexity of string matching algorithms for filtering whole term collection to build the candidate keyword set is O(mN) and that of IR filtering is reduced to O($m$log$N$). Even for general PCs, if the IR technique is applied, automated in-text keyword tagging shows tagging times of 0.5~2.0 seconds for 5~20kb (usually the length of a scientific article falls in this range) of texts and a keyword dictionary of 400 thousand entries. Using high end servers, IR-based in-text keyword tagging is supposed to be practical in many information services.

In the scope of this paper, recall and precision were not considered due to lack of test collections for in-text keyword tagging. From our qualitative analysis, it is expected that recall is high for pattern matching algorithms since substring can be selected as candidate keywords, while precision is high for IR-based keyword tagging since candidate keyword selection is based on token-based comparison between input text and keyword collection. We hope that we can build a test set for in-text tagging in the near future and evaluate our method quantitatively.

The IR-based automated in-text keyword tagging



(a) BMH algorithm



(b) IR filtering

Fig. 5. Detailed view of dictionary size vs. in-text tagging time for (a) BMH algorithm and (b) IR filtering (input text length = 5kb)

method has been successfully applied to a web site for in-text highlighting and hyperlinking of documents in the Animal Picture Archive[†††]. As shown in this example, IR-based in-text keyword tagging can be applied to on-line cross-referencing. Also, we hope that our method can be applied even to off-line keyword tagging tools for XML compilations, generation of cross-references in encyclopedia articles, and HTML documents transformed automatically from plain text collection [7-10]. In addition to these kinds of in-context cross-referencing (XRIC), we hope that our method also can be applied to out-of-context cross-referencing (XROC) such as automatic extraction of major keywords and building book indexes.

Acknowledgment

The first author would like to especially thank Changmin Kim and Jieun Chong for supporting this work.

Reference

[1] Sergey Brin and Lawrence Page, "The Anatomy of a Large-scale Hypertextual Web Search Engine," In Computer Networks and ISDN Systems: *Proceeings of the Seventh International World Wide Web Conference*, Volume 30(1-7):107-117, Apr. 1998.

[2] Hyeon Kim, "Handling XML documents in Hypertext Compilation of the Encyclopedia of Korean Local Culture," *Human Contents*, 9:91-123, 2007.

[3] Jihong Zeng and Peter A. Bloniarz, "From Keywords to Links: An Automatic Approach," In *Proceedings: International Conference on Information Technology: Coding and Computing* (*ITCC'04*), Vol.1, pp.283-286, Las Vegas, Nevada, USA, Apr. 2004.

[4] R. Nigel Horspool, "Practical Fast Searching in Strings," *Software*: *Practice and Experience*, 10(6):501-506, 1980.

[5] William B. Frakes and Ricardo Baeza-Yates, "Information Retrieval: Data Structures & Algorithms," Prentice-Hall, 1992.

[6] Jinsuk Kim, Du-Seok Jin, Yusoo Choi, Chang-Hoo Jeong, Kwangyoung Kim, Sung-Pil Choi, Minho Lee, Min-Hee Cho, Ho-Seop Choe, Hwa-Mook Yoon, and Jeong-Hyun Seo, "Toward DB-IR Integration: Per-Document Basis Transactional Index Maintenance," In *Proceedings: The 6th International Conference on Advanced Language Processing and Web Information Technology* (*ALPIT'07*), Vol.6, pp.452-462, Luoyang, Henan, China, Aug. 2007.

[7] Hsin-Chang Yang and Chung-Hong Lee, "A Text Mining Approach for Automatic Construction of Hypertexts," *Expert Systems with Applications*, 29: 723-734, 2005.

[8] Robert J. Glushko, "Transforming Text into Hypertext for a Compact Disc Encyclopedia," *ACM SIGCHI Bulletin*, 20:293-298, 1989.

[9] Luc Goffinet and Monique Noirhomme-Fraiture, "Automatic Cross-referencing of HCI Guidelines by Statistical Methods," *Interacting with Computers*, 12(2):161-177, 1999.

[10] Airi Salminen, Jean Tague-Sutcliffe, and Charles McClellan, "From Text to Hypertext by Indexing," *ACM Transactions on Information Systems*, 13(1):69-99, 1995.

[11] Jakob Voβ, "Tagging, Folksonomy & Co-Renaissance of Manual Indexing", In *Proceedings*: *The 10th International Symposium for Information Science*, pp.234-254, Cologne, Germany, 2007.

[12] Gilad Mishne, "AutoTag: A Collaborative Approach to Automated Tag Assignment for Weblog Posts", In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pp.953-954, New York, USA, 2006.

[13] Sanjay Sood, Sara Owsley, Kristian Hammond and Larry Birnbaum, "TagAssist: Automatic Tag Suggestion for Blog Posts", In *Proceedings: International Conference on Weblogs and Social Media* (*ICWSM 2007*), Colorado, USA, 2007.

[14] Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme, "Tag Recommendations in Folksonomies", In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases* (*PKDD 2007*), pp.506-514, September 17-21, 2007, Warsaw, Poland.

[15] Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C. Lee Giles, "Real-time Automatic Tag Recommendation", In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (*SIGIR'08*), pp.515-522, July 20-24, 2008, Singapore, Singapore.

[16] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien, "SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation", In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)* Budapest, Hungary, 2003.

[17] Paul - Alexandru Chirita, Stefania Costache, Wolfgang

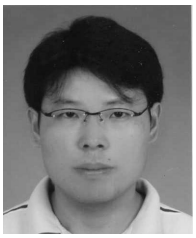
[†††] http://www.animalpicturesarchive.com

Nejdl, and Siegfried Handschuh, "P-TAG: Large Scale Automatic Generation of Personalized Annotation Tags for the Web", In *Proceedings of the 16th international conference on World Wide Web*, pp.845-854, May 08-12, 2007, Banff, Alberta, Canada.

[18] Technion Grigory Begelman, Citrin I. Philipp Keller, and Rawsugar Frank Smadja. "Automated Tag Clustering: Improving Search and Exploration in the Tag Space". In *Collaborative Web Tagging Workshop at WWW2006*, Edinburgh, Scotland, 2006. Online: http://www.rawsugar.com/www2006/20.pdf.

[19] Scott A. Golder and Bernardo A. Huberman, "Usage Patterns of Collaborative Tagging Systems", *Journal of Information Science*, 32(2): 198-208, 2006.

Jinsuk Kim
He is a Senior Researcher of the Department of Information Technology Research at the Korea Institute of Science & Technology Information – South Korea. He earned a M.Sci in Biology and M.Eng. in Computer Science from the Korea Advanced Institute of Science & Technology (KAIST) – Korea. His research interests lie in information retrieval, automated text categorization, bioinformatics and DB-IR integration. He has authored numerous scientific articles.



Du-Seok Jin
He is a Senior Researcher of the Department of Information Technology Research at the Korea Institute of Science & Technology Information – South Korea. He earned a M.Eng. in Computer Science from the Chonbuk National University – Korea. He is currently a Ph.D candidate in Computer Science at Pai-Chai University – Korea. His research interests lie in information retrieval, storage engine on solid state disk and DB-IR integration. He has authored numerous scientific articles.

KwangYoung Kim
He is a Senior Researcher of the Department of Information Technology Research at the Korea Institute of Science & Technology Information – South Korea. He earned a M.Eng. in Computer Science from Busan National University – Korea. He is currently a Ph.D candidate in Library and Information Science at Chungnam National University – Korea. His research interests lie in information retrieval, search algorithms and knowledge base. He has authored numerous scientific articles.

Ho-Seop Choe
He is a Senior Researcher of the Department of Information Technology Research at Korea Institute of Science & Technology Information – South Korea. He earned a Master degree in Korean language from the Kyeong-Nam University – Korea., and a Ph.D. in Computer Science from Ulsan University – Korea, respectively. His research interests lie in intelligent word network and knowledge base. He has authored numerous scientific articles, has served on technical committees for a number of international conferences and has reviewed articles for many journals, conferences and workshops.