A Comparative Study Weighting Schemes for Double Scoring Technique

Tanakorn Wichaiwong *Member, IAENG* and Chuleerat Jaruskulchai

Abstract—**In XML-IR systems, the weighting algorithm plays an important role cause of it greatly affects the precision and recall results of the retrieval systems. Term weight algorithm is widely applied into retrieval models. Since, we developed a XML information retrieval system by using MySQL and Sphinx, namely MEXIR, and extended indices' scheme to handle parameter tuned weight, which we call Double Scoring function. This function has separated the content into two indices by using the XML structure. Thus, we have to investigate weighting schemes and performed a comparative study of Sphinx's weighting schemes processing on MEXIR. Our objective of the study was to find out the appropriate features to achieve the effectiveness of XML Retrieval. The experiment results show the use of BM25 function on leaf-node indices and term frequency on selected weight indices performs better than other methods measured by INEX evaluations.**

***Index Terms*— XML Retrieval, Ranking Strategies, Sphinx, Implementation**

I. Introduction

THE weighting schemes of information retrieval have been widely studied. Many researchers have been studied base on different weighting functions. We developed XML [1] information retrieval system by using MySQL [2] and Sphinx [3, 4], namely MEXIR [5], and extended indices scheme to handle parameter tuned weight, namely Double Scoring function [6]. For this purpose, our study addressing on the comparison of various term weighting base on Sphinx's methods and contribution of weighting schemes area of understanding features that influence the automatic indexing potential of terms.

This paper is organized as follows; Section 2 reviews related works. Section 3 explains our experiments, conclusions and further work are drawn in Section 4.

Manuscript received April 20, 2011; revised May 30, 2011. This work was supported in part by the Graduate School of Kasersart University.

Wichaiwong T. He is currently pursuing the Ph.D. degree in Kasetsart University, Bangkok, Thailand. He current research interests include the area of Information Retrieval, XML Retrieval and XML Query Language and Database. Thailand (phone: 6687-696-1333; e-mail: g5184041@ku.ac.th).

Jaruskulchai C. She received her Bachelor of Education from Chulalongkorn University, and Master of Science in Applied Science in the area of computer Science from National Institute of Development Administration in 1978, and Ph.D. from George Washington University, USA, in 1998. Currently, she served as the Chair Ph.D. program in computer Science, Department of Computer Science, Kasetsart University, Bangkok, Thailand. (e-mail: fscichj@ku.ac.th).

II. Related Works

A. Sphinx Full Text Search Engine Overview

Sphinx [3, 4] is a Full Text Search (FTS) engine that provides fast, size efficient and relevant full-text search functions to other applications. The only kind of collection updating it supports is appending new documents. Document updates and deletions require a complete index rebuild. Sphinx's index structure is very simple. There are two files, a term dictionary and the inverted list. The term dictionary contains for each term an offset into the inverted list file and some statistics. The inverted list file is simply a list of all occurrences for the terms, with no empty space. Reading the inverted list for a term thus requires a maximum of one disk seek.

Sphinx has two types of weighting functions:

- Phrase rank: based on a length of the longest common subsequence (LCS) of search words between the document body and query phrases. So if there's a perfect phrase match in some document then its phrase rank would be the highest possible, and equal to query words count.
- Statistical rank: based on classic BM25 function [7], which only takes word frequencies into account. If the word is rare in the whole database, it receives more weight. Final BM25 weight is a floating point number between 0 and 1.

Sphinx has seven types of search modes:

- MATCH ALL: the final weight is a sum of weighted phrase ranks and matches all query words.
- MATCH ANY: the final weight is a sum of weighted phrase ranks and matches any of the query words.
- MATCH PHRASE: the final weight is a sum of weighted phrase ranks, and matches query as a phrase, requiring the perfect match.
- MATCH_BOOLEAN: matches query as a Boolean expression
- MATCH_EXTENDED: the final weight is a sum of weighted phrase ranks and BM25 weight, multiplied by 1000 and rounded to integer.
- MATCH_EXTENDED2: matches query using the second version of the extended matching mode.
- MATCH_FULLSCAN: matches query, forcibly using the "full scan" mode, any query terms will be ignored, such that filters, filter-ranges and grouping will still be applied, but no text-matching.

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

B. Double Scoring on BM25 (BM25W)

The leaf node indexing is closest to traditional information retrieval since each XML node is a bag of words of itself, and can be scored as ordinary plain text document then we calculate the leaf element score of its context using BM25 of Sphinx as following;

$$LeafScore(e,Q) = \sum_{t \in Q} W_t * \frac{(k_1 + 1) * tf_e}{k_1 * ((1-b) + b * \frac{len(e)}{avel}) + tf_e} \quad (1)$$

$$W_t = \frac{log\left[\frac{N - e_t + 1}{e_t}\right]}{log[N+1]} \quad (2)$$

Note that;
LeafScore(e, Q) measures the relevance of element e in leaf-node indices to a query Q.
W_t is the inverse element frequency weight of term t.
tf_e is the frequency of term t occurring in element e.
len(e) is the length of element e.
avel is the average length of elements in whole collection.
N is the total number of an element in the collection.
e_t is the total element of a term t occur.
k1 and *b* are used to balance the weight of term frequency and element length.

Suppose we have *n* mixed content nodes in given collection *C*. Given a weight W_{nf} for each element *n* this contributes to a given element's weight store in Selected Weight (SW) indices, and then these indices is also closest to traditional information retrieval since each mixed content node is a bag of words of itself, and can be compute the weight for each field as ordinary plain text document, and then we calculate the SW using BM25 as follows;

$$W_{nf} = SW(e,Q)$$

$$SW(e,Q) = \sum_{t \in Q} W_t * \frac{(k_1 + 1) * tf_e}{k_1 * ((1-b) + b * \frac{len(e)}{avel}) + tf_e} \quad (3)$$

Note that;
W_{nf} is the field weight of mixed content elements in Selected Weight indices.
SW(e, Q) measures the relevance of element e in Selected Weight indices to a query Q.

Given a query Q, we run the query in parallel on each index, and then integrate the double scoring function by using the weight from *SWRelList* of SW indices apply to each *LeafNodeRelList* result set from Leaf-Node indices. The weighting for each element in each *LeafNodeRelList* result set is linear combination by *SWRelList* when the prefix of result set is same as the *SWRelList* path, and then the new score for each *LeafNodeRelList* list can compute BM25W as follows;

$$BM25W(e,Q) = \sum_{t \in Q} LeafScore(e,Q) * SW(e,Q) \quad (4)$$

Note that;
BM25W(e, Q) measures the relevance of element e to a query Q.

C. MEXIR System Overview

The More Efficient XML Information Retrieval (MEXIR) is a base on leaf-node indexing scheme and uses a relational DBMS as a storage back-end. We discussed the schema setup using the MySQL and the full-text engine Sphinx using the MySQL dumps.

For the initial step, we consider a simplified XML data model but disregard any kind of Meta mark-up, including comments, links in the form of XLink or ID/IDRef and attributes. In figure 2, depicts the overview of XML retrieval system. The main components of the MEXIR retrieval system are as follows.

- When new documents are entered, the ADXPI Indexer more details as discuss in section E; parses and analyzes the tag and content data to classify indices.
- The Sphinx is used to build the indices.
- The score sharing function more details as discuss in D, is used to assign parent scores by sharing scores from leaf nodes to their parents.
- The double scoring function is used to adjust the leaf nodes score by linear combination.

We used four types of search modes in Sphinx as show in Table I.

TABLE I. THE SPHINX SEARCH MODES

MODE	Description
ANY	The final weight is a sum of weighted phrase ranks and matches ***any*** of the query words.
ALL	The final weight is a sum of weighted phrase ranks and matches ***all*** query words
PHRASE	The final weight is a sum of weighted phrase ranks, and matches query as a ***phrase***, requiring the perfect match.
EXTENDED	The final weight is a sum of weighted phrase ranks and ***BM25*** weight, multiplied by a thousand and rounded to integer.

D. Score Sharing Function

In previous reports [8], we compute the scores of all elements in the collection that contain query terms. We must consider the scores of elements by accounting for their relevant descendents. The scores of retrieved elements are now shared between leaf node and their parents in the document XML tree according to the following scheme.

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$$Score(PNode) \leftarrow Score(PNode) + [(LeafScore) * \beta^n] \quad (5)$$

Note that;
PNode is a current parent node.
β is tuning parameter.

If {0 – 1}, then preference is given to the leaf node over the parents.

Otherwise, preference should be given to the parents.
n is the distance between the current parent node and the leaf node.

E. Absolute Document XPath Indexer

In previous reports [9], a single inverted file can hold the entire reference list, while the suitable indexing of terms can support the fast retrieval of the term-inverted lists. To control overlap and reduce the cost of joined on DBMS; we used the Absolute Document XPath Indexing (ADXPI) scheme to transform each leaf element level into a document level.

```
<?xml version="1.0"?>
<article id = "1">
    <title>xml</title>
    <body>xml
        <section>retrieval
            <title>xml</title>
            <p>information</p>
            <p>retrieval</p>
        </section>
    </body>
</article>
```

Figure 1. The Example of XML Element

In figure 1, depicts the example of the XML element. For instance, take a document named x1; we can build an index using the ADXPI expression to identify a leaf XML node that has text contained within the document, relative to document and its parents are in Table II and Table III.

TABLE II. EXAMPLE OF SW INVERTED FILE.

Term	Inverted List
Xml	x1/article[1]/body[1]
Retrieval	x1/article[1]/body[1]/section[1]

TABLE III. EXAMPLE OF LEAF-NODE INVERTED FILE.

Term	Inverted List
Xml	x1/article[1]/title[1], x1/article[1]/body[1]/section[1]/title[1]
1	x1/article[1]/@id[1]
Information	x1/article[1]/body[1]/section[1]/p[1]
Retrieval	x1/article[1]/body[1]/section[1]/p[2]

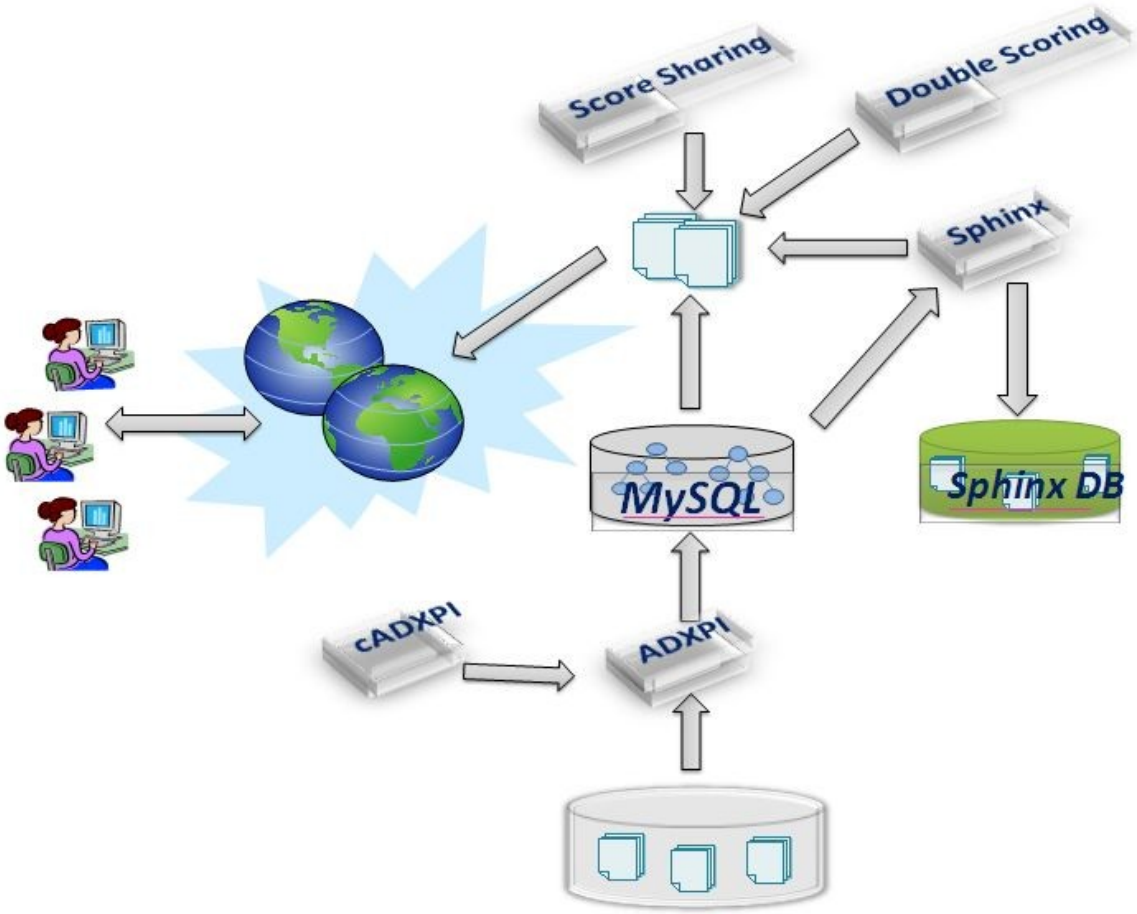

Figure 2. MEXIR System Overview

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

III. Experiment Setup

In this section, we present and discuss the results that were obtained at INEX collections. We performed with the Wikipedia collection. This experiment was done on Intel Pentium i5 4 * 2.79 GHz with the memory of 6 GB, Microsoft Windows 7 Ultimate 64-bit Operating System and using Microsoft Visual C#.NET 2008 for develop system.

A. INEX Collection Tests

The document collections are from the INEX-Wikipedia 2006 XML Corpus for English Wikipedia from early 2006 [10] contains 659,338 Wikipedia articles; the total size is 4.6 GB without images and 52 million elements. On average, an article contains 161.35 XML nodes, whereas the average depth of a node in the XML tree of a document is 6.72. The INEX-Wikipedia 2009 [11] collection was created from the October 8, 2008 dump of the English Wikipedia articles and incorporates semantic annotations from the 2008-w40-2 version of YAGO. It contains 2,666,190 Wikipedia articles and has a total uncompressed size of 50.7 GB. After that, our system uses them in the experiments.

B. INEX Evaluations

Our evaluation is based on the main INEX measures [12]. The main ranking of INEX evaluation is based on iP[0.01] instead of the overall measure MAiP, allowing to emphasize the precision at low recall levels. Our experiment targets for CO Task only, the system accepts CO queries, which are terms enclosed in <title> tag. Then, the Focused Task only remains in INEX 2008 and 2010 topics. Thus the system is evaluated on only Focused Task by inex_eval that tool provided by INEX.

C. Experiment Results and Discussion

In this section, we present the results of evaluation of the MEXIR. We tuned parameters using INEX-2005 Adhoc track evaluation scripts distributed by the INEX organizers. The total number of leaf nodes is 2,500 and the β parameter is set to 0.10 [4], which is used to compute the score sharing function. In order to evaluate the sensitivity of the evaluation, we have used the entire Sphinx match mode values for each index including MATCH ANY (ANY), MATCH ALL (ALL), MATCH PHRASE (PHRASE), and MATCH EXTENDED (EXTEND). We have use the columns indicate to Leaf-Node indices and the rows indicate to SW indices. As such, we report the effectiveness of our system on INEX collections as follows:

TABLE IV. THE EFFECTIVENESS ON IP[0.01] OF INEX-2008 FOCUSED TASK

MODE	ANY	ALL	PHRASE	EXTEND
ANY	0.4419	0.4386	0.4386	0.417
ALL	0.484	0.4751	0.4751	0.4768
PHRASE	0.4595	0.4544	0.4544	0.4514
EXTEND	**0.6499**	**0.6499**	**0.6499**	0.5678

TABLE V. THE EFFECTIVENESS ON MAIP OF INEX-2008 FOCUSED TASK

MODE	ANY	ALL	PHRASE	EXTEND
ANY	0.0961	0.096	0.096	0.0907
ALL	0.0854	0.0835	0.0835	0.0829
PHRASE	0.0870	0.0868	0.0868	0.0868
EXTEND	**0.1828**	0.1827	0.1827	0.1631

TABLE VI. THE EFFECTIVENESS ON IP[0.01] OF INEX-2010 FOCUSED TASK

MODE	ANY	ALL	PHRASE	EXTEND
ANY	0.3285	0.3144	0.3284	0.2791
ALL	0.2469	0.2432	0.2468	0.2463
PHRASE	0.2262	0.2261	0.2261	0.2256
EXTEND	**0.3909**	**0.3909**	**0.3909**	0.3769

TABLE VII. THE EFFECTIVENESS ON MAIP OF INEX-2010 FOCUSED TASK

MODE	ANY	ALL	PHRASE	EXTEND
ANY	0.0619	0.0629	0.0624	0.0615
ALL	0.0500	0.0535	0.0500	0.0499
PHRASE	0.0570	0.0570	0.0570	0.0570
EXTEND	**0.0750**	0.0749	0.0749	0.0728

The performance of different Sphinx's search features is evaluated. Tables IV and Table V show the results obtained by the BM25W ranking functions on INEX-Wikipedia 2006. Table VI and Table VII on INEX-Wikipedia 2009 collection more details are following:

The run BM25W obtained the highest scores for INEX-Wikipedia 2006 on 2008 topics is the MATCH EXTENDED on leaf node indices and MATCH ANY on SW indices as follows: 0.6499 at iP[0.01] and 0.1828 at MAiP respectively. The run BM25W obtained the highest scores for INEX- Wikipedia 2009 on 2010 topics is the MATCH EXTENDED on leaf node indices and MATCH ANY on SW indices as follows: 0.3909 at iP[0.01] and 0.0750 at MAiP respectively.

Due to the BM25 weight is not able to benefit from information contained in the fields with less text. Then the experiment results of Term Frequency on SW index have performed better than other weighting methods.

IV. Conclusions

Due to the ever increasing information available electronically, their size is growing rapidly. The widespread use of XML documents in digital libraries led to the development of information retrieval (IR) methods specifically designed for XML collections. Most traditional IR systems are limited to whole document retrieval; however, since XML documents separate content and structure, XML-IR systems are able to retrieve the relevant portions of documents. Therefore, users who utilize an XML-IR system could potentially receive highly relevant and highly relevant and precise material.

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

In this paper, we performed a comparative study of Sphinx's search modes processing on MEXIR. Our experiment shows that the Sphinx search mode used is the BM25 function of MATCH EXTENDED on leaf node indices and the term frequency of MATCH ANY on Selected Weight indices performs better than other methods measured by INEX evaluations on iP[0.01] and MAiP.

In our future work, we plan to study how to make inferences regarding structural aspects based on CAS queries.

REFERENCES

[1] Extensible Markup Language (XML) 1.1 (Second Edition). http://www.w3.org/TR/xml11/

[2] MySQL Full-Text Search Functions, Available at http://dev.mysql.com/

[3] Sphinx Open Source Search Server, Available at http://www.sphinxsearch.com/

[4] Aksyonoff, A. Introduction to Search with Sphinx, O'Reilly Media. 2011.

[5] Wichaiwong T. and Jaruskulchai C., "MEXIR: An Implementation of High-Speed and High-Precision XML Information Retrieval," The 3rd International Conference on Machine Learning and Computing, Singapore, February 26-28, 2011.

[6] Wichaiwong T. and Jaruskulchai C., "XML Retrieval More Efficient Using Double-Scoring Scheme," The 9th International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX-2010), pages 292–298, 2010.

[7] Robertson, S.E. et al, Okapi at TREC. The Proceedings of the 3rd Text REtrieval Conference, 1995. pp. 109-126.

[8] Wichaiwong T. and Jaruskulchai C., "A Simple Approach to Optimize XML Retrieval," The 6th International Conference on Next Generation Web Services Practices, Goa, India, November 23-25, 2010.

[9] Wichaiwong T. and Jaruskulchai C., "XML Retrieval More Efficient Using ADXPI Indexing Scheme," The 4th International Symposium on Mining and Web, Biopolis, Singapore, March 22-25, 2011.

[10] Denoyer L. and Gallinari P., 2006. The Wikipedia XML Corpus. SIGIR Forum, pp. 64–69.

[11] Schenkel R., Suchanek F. M., and Kasneci G., YAWN: A semantically annotated Wikipedia XML corpus. In 12. GI-Fachtagung f¨ur Datenbanksysteme in Business, Technologie und Web (BTW 2007), pages 277–291, 2007.

[12] Kamps, J. Pehcevski, J. Kazai, G. Lalmas, M. and Robertson, S. 2007. INEX 2007 evaluation measures. In 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, Selected Papers, 2008.

ISBN: 978-988-18210-9-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)