Computing and Informatics, Vol. 32, 2013, 411–440

A DOUBLE SCORING METHOD FOR XML ELEMENT RETRIEVAL

Tanakorn Wichaiwong, Chuleerat Jaruskulchai

Department Of Computer Science
Faculty of Science
Kasetsart University
Bangkok Thailand
e-mail: {g5184041, fscichj}@ku.ac.th

Communicated by Jacek Kitowski

Abstract. Efficient retrieval of XML elements and documents is essential in the effective application of the XML format. The ranking function BM25F is composed of several document fields with potentially different degrees of importance; these fields are known as selected fields that give substantial improvements over the baseline BM25. The BM25F function has performed well in past evaluations; however, there are issues that require additional attention. In the first instance, which elements should be treated as fields? Secondly, what is an appropriate weight for each field? Previously, document fields were selected manually, and the weight for each chosen field was tuned before being assigned. Two automatic methods are introduced in this paper that enable the extraction of fields in document-centric XML documents and the assignment weights to the selected fields. Our experiments show an improvement of up to 28 % over BM25, and up to 15 % over BM25F at iP[0.01] based on INEX evaluations.

Keywords: Ranking strategies, indexing units, XML retrieval, BM25F

1 INTRODUCTION

The widespread use of Extensible Markup Language (XML)[1] documents in digital libraries has led to the development of information retrieval (IR) methods specifi-

[1] http://www.w3.org/TR/xml11/

cally designed for XML collections. Most traditional IR systems are limited to whole document retrieval. In contrast to document retrieval, there are no pre-defined fixed retrieval units in XML retrieval. XML documents separate content and structure; XML-IR systems can retrieve relevant portions of documents. A document is indexed by a set of terms, and each term has associated weighting functions. These term statistics are employed by the retrieval algorithm to estimate to estimate the relevance of the document. Thus, the weighting function plays an important role because it greatly affects the precision and recall results of the retrieval systems.

According to the Initiative for the Evaluation of XML Retrieval (INEX)[2] [11] studies, information about document structure can be used to boost terms that occur in any specific document field or XML element. For example, a term occurring in element "title" might be more important than a term appearing in the "section" or "paragraph" elements. These heuristics have been applied with success to optimize IR system which we call BM25F [34, 4, 20, 35, 17]. The BM25F function is an extension of the BM25 ranking retrieval function that is adapted to score field documents. BM25F is composed of several document fields or elements with potentially different degrees of importance. However, there are questions that have not been appropriately addressed in the past. Previously, field documents have been selected manually by an expert. The weight was tuned before it was assigned. This exacerbated pre-processing complexity with respect to cost and time and encouraged heterogeneous collections [31, 32, 37]. Therefore, we investigated two research issues in this article:

1. Which elements should be considered fields?
2. What is an appropriate weight for each field?

We propose two automatic methods; the selected fields are automatically chosen using mixed content elements, and the elements are not tuned for each selected field. Rather, the weights of the fields are tuned for effective IR system.

This paper is organized as follows. Section 2 reviews related work. Section 3 presents a method of addressing the problems. Section 4 explains our approaches to these issues, and Section 5 shows the implementation of our system. Section 6 presents the experimental setup, results and discussion. Section 7 provides conclusions and recommendations for further work.

2 RELATED WORK

In this section, we provide an overview of previous XML research that has influenced this article.

[2] https://inex.mmci.uni-saarland.de/

2.1 XML Indexing Methods

The basic XML data model is a labeled, ordered tree. Figure 1 shows the data tree of an XML document based on the node-labeled model.

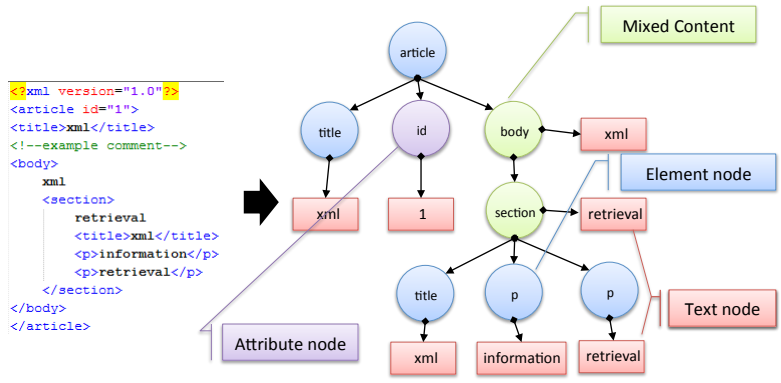

Fig. 1. The example of an XML element tree

Classical retrieval models have been adapted to XML retrieval. Several indexing strategies [15] have been developed in XML retrieval, as shown in Figure 2.

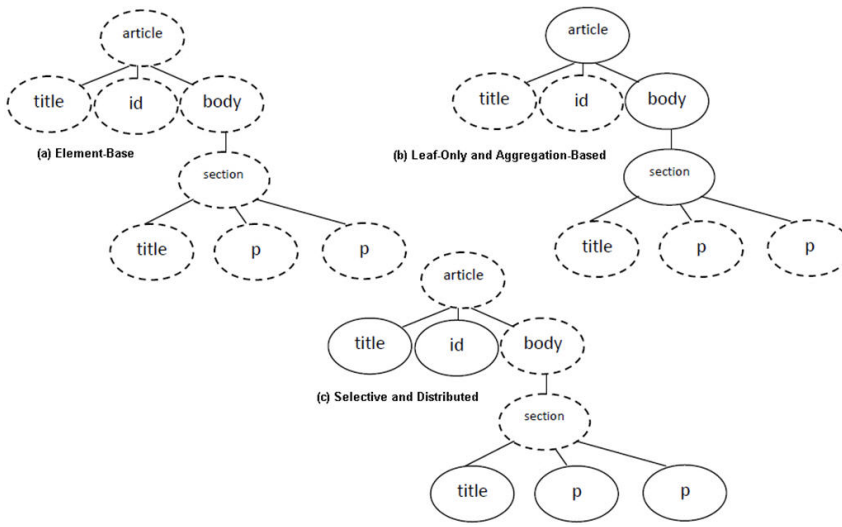

Fig. 2. Illustrations of some of the indexing strategies

Element-Base indexing [25, 12, 13] allows each element to be indexed on the basis of both direct text and the text of descendants. This strategy has a major drawback

in that it is highly redundant. Text occurring at the n^{th} level of the XML logical structure is indexed n times and thus requires more index space. This strategy is illustrated in Figure 2 a), where all elements are indexed. Leaf-Only indexing [8, 9, 10] allows indexing of only leaves through element or elements directly related to text. This strategy addresses the redundancy issues noted above. However, the propagation algorithm for the retrieval of non-leaf elements requires a certain level of efficiency. This strategy is illustrated in Figure 2 b), where the leaf elements are indexed. Aggregation-Based indexing [25, 26, 27, 28] uses the concatenated text of an element to estimate a term statistic. This strategy has been used to aggregate term statistics directly on the basis of the text and its descendants. This is illustrated in Figure 2 b), where the leaf elements are indexed. Selective indexing [8, 21] involves eliminating small elements and elements of a selected type; this strategy is illustrated in Figure 2 c), where only semantic elements are indexed. Distributed indexing [21, 22, 23, 24] is separately created for each type of element in conjunction with the selective indexing strategy, as shown in Figure 2 c). The ranking model runs each index separately and retrieves ranked lists of elements. These lists are merged to provide a single rank across all element types. To merge lists, normalization is performed to take into account the variation in elements size across the different indices such that scores across indices are comparable.

2.2 XML Ranking Methods

Ranking schemes depend on the chosen indexing strategies and specific mechanisms, such as propagation and aggregation in which only leaf elements are indexed. The various approaches to XML ranking strategies serve as models for traditional document retrieval. The majority of these ranking methods maintain index and rank elements with or without structural constraints, according to an element relevance to a given query. The relevance score estimated from the statistics of a term at the element level or document level is as follows.

Prior works [34, 4, 20, 35, 17] present BM25F as an extension of the baseline BM25 [33, 35] scoring function that is adapted to score field documents. Using the BM25F scheme presented in [17], an element score is computed as follows:

The normalized of term frequency is obtained by first performing the length on term frequency $W_{e,f,t}$ of a term t in a field f in an element e.

$$W_{e,f,t} = \frac{tf_{e,f,t}}{1 + b_f * \left(\frac{len_{e,f}}{avglen_f} - 1\right)} \tag{1}$$

Note that:

- b_f is a parameter.
- $len_{e,f}$ is the length of field f in element e.
- $avglen_f$ is the average length of elements in the entire collection after multiplying the normalized term frequency $W_{e,f,t}$ by field weight W_f.

$$tf_{e,t} = \sum_{f} W_f * W_{e,f,t} \tag{2}$$

$$BM25F(e,q) = \sum_{t \in q \cup e} W_t * \frac{tf_{e,t}}{k_1 + tf_{e,t}} \tag{3}$$

Note that:

- $BM25F(e,q)$ measures the relevance of element e to query q.
- q is a set of query terms.
- $tf_{e,f}$ is a weighted normalized term frequency.
- k_1 is a parameter to control the non-linear growing term frequency function.
- W_t is the inverse document frequency weight of term t.

Element scoring [12, 13, 18] is based on language models and employs element-based indexing. Given a query q, terms t_i for each element e and its corresponding element language model $\ominus_e$, the element e is ranked as follows:

$$P(e \mid q) = P(e) * P(q \mid \ominus_e). \tag{4}$$

Note that:

- $P(e)$ is the probability of relevance for element e.
- $P(q \mid \ominus_e)$ is the probability of the query q generated by language model $\ominus_e$.

For instance,

$$P(t_1, \cdots, t_n \mid \ominus_e) = \prod_{i=1}^{n} \lambda P(t_i \mid e) + (1-\lambda) P(t_i \mid C) \tag{5}$$

Note that:

- $P(t_i \mid e)$ is estimation of term t_i in element e.
- $P(t_i \mid C)$ is the probability of term t_i in collection C.
- λ is the smoothing parameter, and $length_e$ is the element length, which can be used to set $P(e)$ as follows:

$$P(e) = \frac{length_e}{\sum_c length_e}. \tag{6}$$

This strategy is based on element-based indexing; therefore, it also has the problem of term redundancy.

Score propagation [8, 9, 10] is used to rank elements based on leaf-node indexing. Scoring is propagated upward to ancestors. The resulting relevance score for each element is a weighted accumulation of ranking scores of the element's children. This strategy was presented by the Gardens Point XML-IR (GPX) [8, 9, 10], a propagation method that was proposed as a bottom-up scheme (BUS) [40]. For example, for each element with only one relevant child element, the child should be ranked higher. Otherwise, this element is ranked higher than the child. The assignment of a numeric score to a document given a query can be represented as follows:

$$score(e, q) = D(m)_e * \sum_{e,c} score(e_c, q). \tag{7}$$

Note that:

- $D(m)$ is the smoothing parameter for each element e set as follows.
- If e has one child, then $D(m)_e = 0.49$.
- Otherwise, $D(m)_e = 0.99$.

Score aggregation methods [25, 26, 28] rank elements based on the aggregated representation of the term statistics of its own context with the statistics of the element's children. The effectiveness of the aggregation depends on β parameters, which set the weight for each component and are estimated through learning methods. The assignment of a numeric score to a document for a query can be represented as follows:

$$P(t \mid \ominus_e) = \beta_{own} * P(t \mid \ominus_{e,own}) + \sum_{e,j} \beta_j * P(t \mid \ominus_{e,j}). \tag{8}$$

Note that:

- $\beta_{own} + \sum_{e,j} \beta_j = 1$ the model for the β parameters includes the contribution of each language model in the aggregation. The drawback of this model involves the estimation of this β parameter.

Score merging [21, 22, 23, 24] is adopted as the indexing strategy for different types of elements. Indexing is separately created for each type of element. The ranking model must run each index separately and retrieve ranked lists of elements. These lists are merged to provide a single rank across all element types. To merge lists, normalization is performed to take into account the variation in size of the elements in different indices, so that scores across indices are comparable. The assignment of a numeric score to a document for a query can be represented as follows:

$$score(e, q) = \frac{\sum_{e \in c} W_{t,q} * W_{t,e} * ief_t}{length_e * length_q}. \tag{9}$$

Note that:

- $length_q$ is the length of the query in terms.

- $length_e$ is the length of the element in terms.

Recall that the distribution indexing chosen for this strategy has issues related to indexing maintenance.

RUN ID	**Ranking Method**	**iP**[0.01]
p78-UWatFERBM25F	BM25F	0.6333
p68-I09LIP6Okapi	BM25	0.6141
p10-MPII-COFoBM	BM25 & Proximity	0.6134
p60-UJM-15525	BM25	0.6060
p6-UamsFSsec2docbi100	Element Scoring	0.5997

Table 1. Top 5 participants in the ad hoc track focused task of INEX 2009

Table 1 shows the effectiveness of INEX 2009 evaluations [11] and the "p78-UWatFERBM25F" from the University of Waterloo [16]. This run, which uses a BM25F over two fields – "title" and "body" – has outperformed than other runs; "p6-UamsFSsec2docbi100" is based on element scoring and uses a language model from the University of Amsterdam [18]. "p10-MPII-COFoBM" is based on TopX models that make use of proximity information. It usually improves the effectiveness of search systems [3] and other runs that are based on the baseline BM25 on a standard article index with the b and k parameters tuned [2, 6]. However, there are issues that need more attention. We thus briefly describe the problem statements in next section.

3 PROBLEM STATEMENT

XML documents often contain elements such as INEX collections from IEEE containing fields such as "title", "abs", "bdy", "st", or from INEX-2006 containing fields such as "title", "body", "section", "caption", "collectionlink", "emph2", or from INEX-2009 containing fields such as "article", "sec", "personality", "vehicles", etc. Previous studies [34, 4] have found it beneficial to exploit the document's internal structure to improve the effectiveness in the IR system. In [34, 4] the role of field weight W_f of BM25F (2) is highlighted. Because all weights must be tuned for each selected field, this issue contributes to the weight of document in the BM25F function. The authors show that the tuning values for W_f are all integers, and they tune W_f (atl, abs, st) from $(1, 1, 1)$ to (max, max, max), where max is the limit of increasing value in the experiment using increments of 1. The result shows that the values of $(2356, 4, 22)$ for W_f return the highest average precision score. Along these lines, another study [16] constructed two fields manually, one for "title" and another for "body". The "title" field consists of the concatenation of an article title and any ancestral and current section titles. The "body" field contains the rest of the text in the element. Unfortunately, because the INEX-2006 collection has only one-level of section headings whereas the INEX-2009 collection has subsection headings, and considers only the first level section headings. Thereafter, the values

for W_f $(title, body)$ are tuned from $(1, 1)$ to (max, max) using increments of 0.1. The result shows that the values of $(4.0, 1.2)$ for W_f achieve the highest result on iP[0.01].

For the manually selected fields from the various document design of XML level for each collection, our assumption in using content at the document level implies unfairness for all elements. For instance, the content in element "article[1]/body[1]/section[1]" is not reflected into element "article[1]/title[1]" with respect to the absolute XPath. Rather, it reflects their descendants as follows:

- article[1]/body[1]/section[1]/title[1]
- article[1]/body[1]/section[1]/p[1]
- article[1]/body[1]/section[1]/p[2].

In previous studies [41, 42, 16], the document structure was exploited to improve effectiveness in the IR system. The analytical operations were usually performed manually by experts who increased the pre-processing complexity at the expense of cost and time while encouraging heterogeneous collections [31, 32, 37]. Furthermore, the tuning values for W_f of BM25F function required the training data and evaluation set that were difficult to implement for new and heterogeneous collections.

4 METHODS

In this section, to address the research issues of this paper we just briefly describe two automatic methods. The first method can be used for automatic selection of fields using mixed content elements. The second method implements the scoring function to assign weights to each selected field obtained by the first method.

4.1 Automatic Extraction of Mixed Elements (AutoMix)

XML documents are often divided into two categories; namely data-centric and document-centric documents. The data-centric documents are characterized by a fairly regular structure with no mixed content. The document-centric documents are usually documents that are designed for human consumption. They are characterized by a less regular or an irregular structure and a large amount of mixed content. A content model of the XML element type is *mixed content* when elements of that type may contain text with child elements.

To solve the first problem related to the BM25F function, automatic field selection and a new automatic extraction is proposed. Our approach considers the use of an automatic method to choose selected fields using only mixed content elements. As a result, the introduction of mixed content was necessary. We believe that mixed content elements, and to some extent their descendants, can help in deriving an automatic method to choose selected fields. We propose the use of a method called *AutoMix* for the extraction of mixed content elements. The mixed content elements are separated into new indices known as Selected Weight (SW) as follows.

N: is an element in document D
C_n: is a Child node of N
T: is a Text Node
SW_{index}: is a Selected index
LN_{index}: is a Leaf Node index

```
SELECTMIXCONTENT(Node N)
BEGIN
    IF N contains Child C_n THEN
        IF N contains Text T THEN
            ADD Node N and Text T TO SW_index
        END-IF
        FOREACH Child C_n IN Node N
            SELECTMIXCONTENT(C_n)
        END-FOR
    ELSE
        IF N contains Text T THEN
            ADD Node N and Text T TO LN_index
        END-IF
    END-IF
END.
```

Algorithm 1: AutoMix Algorithm

In the following algorithm description, indentation is used to denote the details of algorithm.

1. Read all nodes N entries from the document D
2. For each node N in the list of No. 1
3. If node N contains child C_n and N contains text T Then
   ADD Node N and Text T TO SW_{index}
   For each child C_n in node N go to 3
   $Otherwise$ go to 4
4. If node N contains text T Then
   ADD Node N and Text T TO LN_{index}
5. When all nodes N from the list are calculated, and then return SW_{index} and LN_{index}

After that we classify mixed content elements. At first, we consider a simplifed XML data model, but we disregard comments, links and attributes. Referring to an example of an XML element tree, we classify tags automatically, and then we build new indices as follows:

Selected Weight Index:

- x1/article[1]/body[1]: "xml"
- x1/article[1]/body[1]/section[1]: "retrieval"

Leaf Node Index:

- x1/article[1]/title[1]: "xml"
- x1/article[1]/body[1]/section[1]/title[1]: "xml"
- x1/article[1]/body[1]/section[1]/p[1]: "information"
- x1/article[1]/body[1]/section[1]/p[2]: "retrieval"

4.2 Double Scoring Function

Previous studies [41, 42] present the extended BM25 function, which is known as compactness of the baseline BM25. However, this strategy is limited in that each tag name must be the same to implement automatic grouping and weight calculation.

We present a new method to reduce parameter tuning, namely, the Double Scoring function [44]. This function is based on new extended indices that store all of the selected fields. Selected Weight indexing is similar to traditional information retrieval because each XML node is a bag of words, and can thus be scored as an ordinary plain text documents. Then, we calculate the Selected Weight index using the BM25 function. After this step, we can compute the element score as follows.

In the following algorithm description, indentation is used to denote the details of algorithm.

1. Set the relevance list L_{rel} to empty list
2. Read the both lists from L_{sw} and L_{ln}
3. For each list SW in the relevance of L_{sw}
   Set $Weight_{sw} = SW_{\text{score}}$
4. For each list LN in the relevance of L_{ln}
   Set $Weight_{ln} = LN_{\text{score}}$
5. If $SW \in LN$ then $ADD\ LN$, $(Weight_{sw} * Weight_{ln})$ to list L_{rel}
   $Otherwise\ ADD\ LN$, $Weight_{ln}$ to list L_{rel}
6. When all lists from L_{sw} and L_{ln} are calculated, then return L_{rel}

We define $Score(e, q)$ is a score for the relevance of a term t of an element e and then we used the baseline BM25 in Sphinx's formula to score the element nodes according to query terms t contained in content conditions as follows:

$$Score(e, q) = W_t * \frac{(k_1 + 1) * tf_{t,e}}{k_1 * \left((1 - b) + b * \frac{len_{(e)}}{avel}\right) + tf_{t,e}}. \tag{10}$$

L_{rel}: is the relevance list
L_{sw}: is the list of relevance from the Selected index
L_{ln}: is the list of relevance from the Leaf Node index

DOUBLESCORING(List L_{sw}, List L_{ln})
BEGIN
  SET L_{rel} := EMPTY
  FOREACH List SW **IN** L_{sw}
    SET $Weight_{sw} := SW_{\text{score}}$
    FOREACH List LN **IN** L_{ln}
      SET $Weight_{ln} := LN_{\text{score}}$
      IF $SW \in LN$ **THEN**
        ADD LN, $Weight_{sw} * Weight_{ln}$ **TO** List L_{rel}
      ELSE
        ADD LN, $Weight_{ln}$ **TO** List L_{rel}
      END-IF
    END-FOR
  END-FOR
END.

Algorithm 2: DoubleScoring Algorithm

Note that:

- $Score(e, q)$ measures the relevance of element e to query q.
- $tf_{t,e}$ 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 the entire collection.
- k_1 and b are used to balance the weight of term frequency and element length.

Then, we compute the inverse element frequency W_t as follows:

$$W_t = \frac{\log\left[\frac{N-e_t+1}{e_t}\right]}{\log(N+1)}. \tag{11}$$

Note that:

- W_t is the inverse element frequency weight of term t.
- N is the total number of an element in the entire collection.
- e_t is the total element of a term t occur.

From Equation (2), we can see that a linear combination of the weighted field frequencies is used instead of the original term frequency in specified fields. We

hypothesize that this method can also be applied to all of the mixed content elements that we propose in the SW indices. Our basic assumption is that an element should be treated like a document.

Suppose we have n mixed content nodes in a given collection C. Given a weight $W_{f,n}$ for each element n, this contributes to a given weight of element stored in the SW indices. These indices are similar to document retrieval because each mixed content node is a bag of words of itself. Therefore, the BM25 weight cannot benefit from the information contained in the fields with less text. Thus, we discard the mixed content elements in the result set while keeping elements under the same indices of leaf-node content, and then we calculate the Selected Weight index using term frequencies (TF) according to [43]. For each weight W_f of the BM25F (2), we multiply each relevance list of Selected Weight by $W_{f,n}$ as follows:

$$W_{f,n} = SW(e, q). \tag{12}$$

Note that:

- $W_{f,n}$ is the field weight of mixed content elements e in the Selected Weight index.

Then, we compute score for each element e as follows:

$$SW(e, q) = \sum_{t \in q} tf_e. \tag{13}$$

Note that:

- $SW(e, q)$ measures the relevance of element e in the Selected Weight index to query q.
- tf_e is the frequency of term t occurring in element e.

We run the query q in parallel on each index (i.e., the SW_{index} and LN_{index}), and then we use the new score for each list to implement the BM25W as follows:

$$\text{BM25W}(e, q) = Score(e, q) * \prod_{i \in n} SW(e_i, q). \tag{14}$$

Note that:

- The $BM25W(e, q)$ measures the relevance of element e to query q.

Following this, we compare the double scoring (BM25W) with the BM25F function as shown in Table 2.

5 IMPLEMENTATION OF XML RETRIEVAL SYSTEM

5.1 XML Retrieval Model

The More Efficient XML Information Retrieval (MEXIR) [45] is based on a leaf-node indexing scheme that uses a relational DBMS as a storage back-end. We discuss the

BM25W	**BM25F**
1. This model has two indices, both SW and Leaf Node index	1. This model has only the Leaf Node index
2. The document fields are automatically chosen using mixed content elements	2. The document fields are selected manually by an expert
3. W_f is the weight automatically assigned at query time	3. W_f is the weight manually assigned in data pre-processing time
4. W_f does not require a tuning parameter	4. W_f requires a tuning parameter
5. W_f does not require training data and an evaluation set	5. W_f requires training data and an evaluation set

Table 2. Comparison between of the BM25W and the BM25F functions

schema setup using MySQL[3] and the full-text engine Sphinx[4] [1] with the MySQL dumps function.

Sphinx has two types of weighting functions:

1. The phrase rank is based on the length of the longest common subsequence (LCS) of search words between the document body and query phrases. If there is a perfect phrase match in some document, then its phrase rank would be equal to the number of query words, which would be the highest possible rank.
2. The statistical rank is based on the classic BM25 function, which only takes word frequencies into account. If a word is rare in the entire database, it receives more weight.

At first, we consider a simplified XML data model, but we disregard Meta markup such as comments, links and attributes. Figure 3 depicts the overview of the XML retrieval system. The main components of the MEXIR retrieval system are as follows:

1. When new documents are entered into the system, the Absolute Document XPath Indexing (ADXPI) [46] indexer parses and analyzes the name of an element and its position to build the inverted lists for each index in this system.
2. The extension XML compression for ADXPI (ecADXPI) compressor analyzes the frequency of each element and its position to build the mapping of dictionary base, where the compressed data is stored into MySQL database.
3. The AutoMix analyzes the tree position for each element to separate content to build the Selected Weight (SW) and Leaf Node indices, which are stored in the MySQL database.
4. The SphinxDB search engine is used to build both indices in the system. The Selected Weight index is based on Term Frequency, and the Leaf Node index is based on the classic BM25 function.

[3] `http://dev.mysql.com/`

[4] `http://www.sphinxsearch.com/`

5. The Score Sharing function is used to assign parent scores by sharing scores from leaf nodes to their parents using a top-down approach.
6. The Double Scoring function is used to adjust the Leaf Node scores based on linear combination.

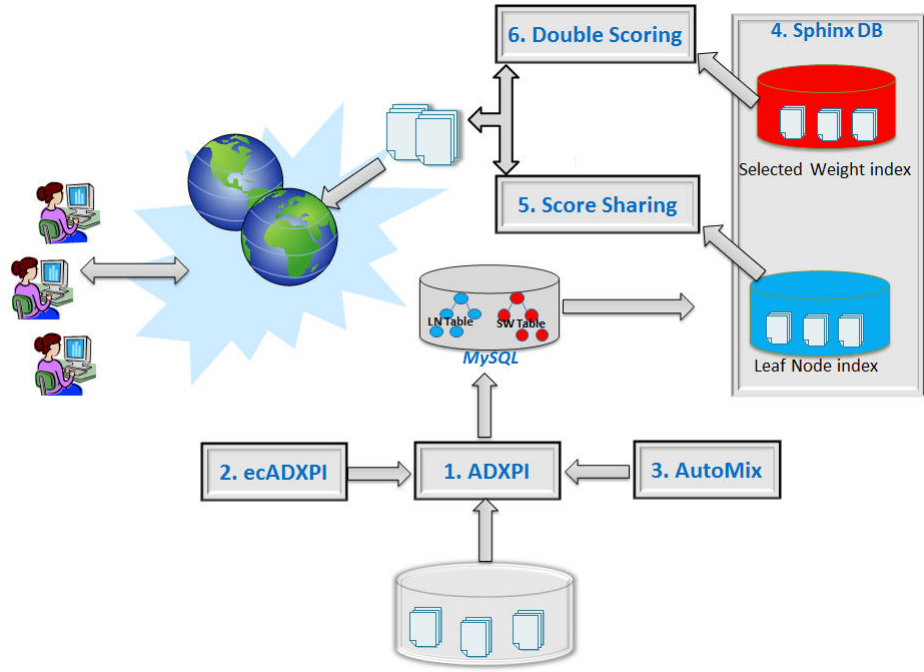

Fig. 3. The MEXIR system overview

5.2 The ADXPI Indexer

According to previous studies [46, 7], a single inverted file can hold the entire reference list, while a suitable indexing of terms can support the fast retrieval of term-inverted lists. To control overlap and reduce DBMS cost, we use the Absolute Document XPath Indexing (ADXPI) scheme to transform each leaf element level into a document level. For instance, consider 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 the document and its parents according to *AutoMix* function as shown in Figure 4 a).

5.3 The Score Sharing Function

As in a previous study [48], we compute the scores of all elements from 2, in the collection that contain query terms. We consider the scores of elements e by accounting for their relevant descendants e_c. The scores of retrieved elements $Score(e, q)$ are now shared between the leaf node and their parents in the document XML tree according to the following scheme.

$$Score(e, q) \leftarrow Score(e, q) + \langle \sum_{e,c} Score(e_c, q) * \beta^n \rangle \tag{15}$$

Note that:

- $Score(e, q)$ is a current parent node.
- $Score(e_c, q)$ is a relevant child of element e.
- β is a tuning parameter.
- $IF\{0 - 1\}$ **THEN** preference is given to the leaf node over the parents.
- $OTHERWISE$, preference is given to the parents.
- n is the distance between the current parent node and the leaf node.

For instance, the query q contains "xml retrieval", consider the LN (for the score sharing, see in No. 5 of Figure 3) and SW indices (for the double scoring see in No. 6 of Figure 3); we assume the score $Score(e, q)$ for each element e is 10 and β is 0.7; the calculations are as shown in Figure 4.

6 EXPERIMENT SETUP

In this section, we present and discuss the results based on the INEX collection. We also present the results of an empirical sensitivity analysis of various parameters performed on a Wikipedia collection. This experiment was performed on Intel Pentium i5 4×2.79 GHz with 6 GB of memory, Microsoft Windows 7 Ultimate 64-bit Operating System and Microsoft Visual C♯.NET 2008.

6.1 INEX Collection Tests

The document collections are from the INEX-IEEE document collection which contains a total of 16 819 articles from 24 IEEE Computer Society journals covering the period 1995–2005. In its canonical form, it is 764 megabytes in size and holds 11 million elements.

The INEX-2006 XML Corpus for English Wikipedia from early 2006 [5] contains 659 338 Wikipedia articles; the total size is 4.6 GB without images, and includes 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.

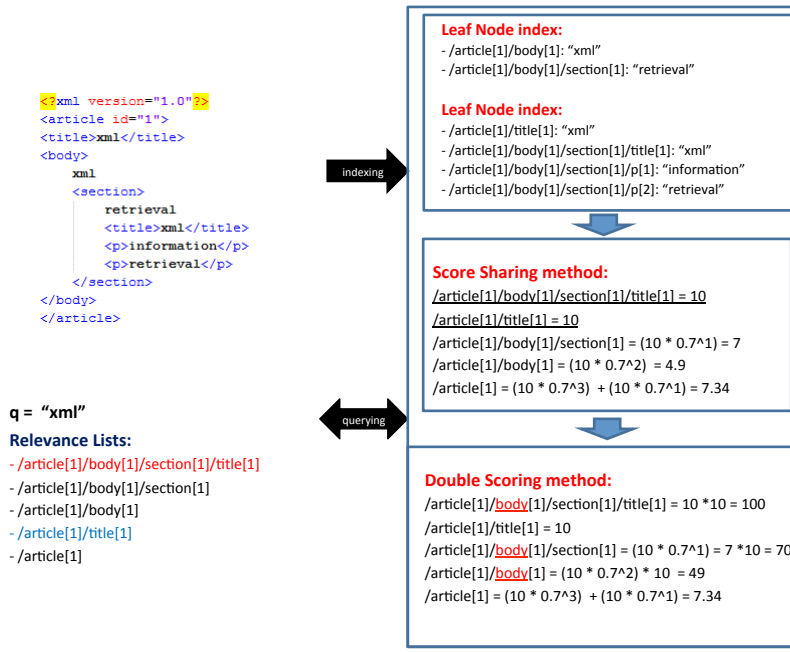

Fig. 4. Illustrations of the query processing

The INEX-2009 [38] collection was created from the October 8, 2008 dump of 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. There are 101 917 424 XML elements of at least 50 characters. Sphinx indexing took 1 minute for INEX-IEEE, 5 minutes for INEX-Wiki06, and 30 minutes for INEX-Wiki09. After indexing, we were able to perform our experiments.

6.2 INEX Evaluation

As for INEX-IEEE effectiveness [39, 30, 36], we refer to the relative and absolute precision values as well as the non-interpolated mean average precision (MAP), which displays absolute (i.e., user-perceived) precision as a function of absolute recall using official relevance assessments provided by INEX. Furthermore, the following, more sophisticated XML-specific metrics were newly introduced for the INEX-IEEE benchmark. The normalized extended Cumulated Gain (CG) metrics are an extension of the Cumulated Gain metrics that consider the dependency of XML elements (e.g., overlap and near-misses) within an evaluation.

As for INEX-Wikipedia effectiveness [29, 19, 14], we refer to the main ranking of INEX evaluation based on iP[0.01] instead of the overall measure mean average

interpolate precision (MAiP), we adopted an evaluation framework that is based on the amount of highlighted text in relevant documents. More formally, let p_r be the document part assigned to rank r returned for a topic q. Let $rsize(p_r)$ be the length of highlighted text contained by p_r in characters. Let $size(p_r)$ be the total number of characters contained by p_r, and let $Trel_q$ be the total amount of highlighted relevant text for topic q. $Trel(q)$ is calculated as the total number of highlighted characters across all documents. A measure at selected precision at rank r is defined as follows:

$$P_r = \frac{\sum_{i=1}^{r} rsize(p_i)}{\sum_{i=1}^{r} size(p_i)}. \tag{16}$$

To achieve a high precision score at rank r, the document parts are retrieved. Recall at rank r is defined as follows:

$$R_r = \frac{\sum_{i=1}^{r} rsize(p_i)}{Trel_q}. \tag{17}$$

To achieve a high recall score at rank r, the document parts retrieved need to contain as much relevant text as possible. An issue with the precision measure P_r given in Equation (16), an iP_x, which calculates interpolated precision scores at recall levels:

$$iP_x = \max_{1 \le r \le |L_q|} P_r \wedge R_r \ge x; \Rightarrow x \le R[|\ L_q\ |] \tag{18}$$

or

$$iP_x = 0; \Rightarrow x > R[|\ L_q\ |]. \tag{19}$$

Note that:

- $R[|L_q|]$ is the recall over all documents retrieved.

Overall performance measure scores are based on the measure of average interpolated precision (AiP). For each topic, the AiP is calculate as follows:

$$AiP = \frac{1}{101} * \sum_{0.00}^{1.00} iP_x. \tag{20}$$

Performance across a set of topics is measured by calculating the mean of the AiP values obtained by the measure for each individual topic, resulting in MAiP. Assuming there are n topics:

$$MAiP = \frac{1}{n} * \sum_{t=1}^{n} AiP_t. \tag{21}$$

Our experiment only targets the CO Task as well as systems that accept CO queries. Note that CO queries are terms enclosed in the <title> tag. After this operation, only the focused task remains in the INEX for 2005, 2008, and 2010 for each collection. Thus, the system is evaluated only using the focused task according to the *inex-eval* and $EvaJ$ tools provided by INEX.

6.3 Experiment Results and Discussion

6.3.1 AutoMix Results

For the evaluation of *AutoMix* algorithm, we chose the 11 916 documents in part-1 of INEX-2009 for our experiment. The size of our test was 208 MB and the average size per document was 18.3 KB. We found that this technique achieved up to 100 % of our requirements. For example, we considered an example of an XML document named "60313" and we then classified the mixed content elements automatically using the AutoMix function. Following that, we built new indices; consider document named "60313" as shown in Figure 5.

Another example of document name "60843" was shown in Figure 6.

Next we considered the data structure of INEX collections by using the AutoMix function and we then constructed both Selected Weight and Leaf Node indices and noted the total number of elements for each INEX collection under analysis. As shown in Table 3, the INEX-IEEE exhibited 8.59 % mixed content, and the average length of the mixed content was 337.05 bytes per element. The INEX-2006 exhibited 32.64 % mixed content elements, and the average length of the mixed context was 15.39 bytes per element. The INEX-2009 exhibited 23.63 % mixed content elements with the average length of the mixed context being 52.83 bytes per element.

Collections	Mixed Node	Leaf Node	Avg. Size (Byte)	%
INEX-IEEE	100 795	1 073 179	337.05	8.59
INEX-2006	2 201 529	4 544 086	15.39	32.64
INEX-2009	4 117 364	11 605 874	52.83	23.63

Table 3. The number of elements on INEX collections

6.3.2 Double Scoring Results

In this section, we present the results used to evaluate our system. In principle, any portion of an XML document can be retrieved, although some portions are more likely to be relevant to the user query.

For the evaluation of the Double Scoring algorithm, we first tuned the parameters of score sharing function using INEX-2005 Ad hoc track evaluation scripts distributed by the INEX organizers. Our tuning approach was such that the sums of all relevance scores were maximized, as shown in Figures 7 and 8. The total number of leaf nodes was 2 500, and the β parameter was set to 0.10, which was used to compute the score sharing function.

Following that, we used the Sphinx parameters for the BM25 where $k_1 = 1.20$ and $b = 0.00$ [47]. To evaluate the sensitivity of the evaluation, we used the entire Sphinx match mode values for each index, including MATCH BOOLEAN (BOOLEAN), MATCH ANY (ANY), MATCH ALL (ALL), MATCH PHRASE (PHRASE), and MATCH EXTENDED (EXTEND). Additional details are provided

```
<?xml version="1.0" encoding="UTF-8"?>
<article>
<name id="60313">Dolfin</name>
<conversionwarning>0</conversionwarning>

<body>

<emph2>For the marine animal see
<collectionlink xlink:type="simple"
xlink:href="9061.xml">Dolphin</collectionlink>
</emph2>

<figure>
<image xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="../pictures/Dolfinlogo.gif" id="157048" xlink:actuate="onLoad"
xlink:show="embed">Dolfinlogo.gif
</image>
<caption>
The<emph3>Dolfin</emph3>
<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple" xlink:href="52888.xml">logo
</collectionlink>
</caption>
</figure>

<p>
<emph3>Dolfin</emph3> is a brand of
<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="223338.xml">swimwear
</collectionlink> that sells
<collectionlink xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:type="simple"
xlink:href="1335181.xml">competitive swimwear
</collectionlink> and supplies.
</p>

<template name="corp stub"></template>

</body>

</article>
```

Selected Weight Index:

- 60313/article[1]/body[1]: ":"
- 60313/article[1]/body[1]/emph2[1]: "For the marine animal see"
- 60313/article[1]/body[1]/p[1]: "is a brand of that sells and supplies."
- 60313/article[1]/body[1]/figure[1]/caption[1]: "The"

Leaf Node Index:

- 60313/article[1]/name[1]: "Dolfin"
- 60313/article[1]/conversionwarning[1]: "Dolfin"
- 60313/article[1]/body[1]/emph2[1]/collectionlink[1]: "Dolfin"
- 60313/article[1]/body[1]/figure[1]/image[1]: "Dolfinlogo.gif"
- 60313/article[1]/body[1]/figure[1]/caption[1]/emph3[1]: "Dolfin"
- 60313/article[1]/body[1]/figure[1]/caption[1]/collectionlink[1]: "logo"
- 60313/article[1]/body[1]/p[1]/emph3[1]: "Dolfin"
- 60313/article[1]/body[1]/p[1]/collectionlink[1]: "swimwear"
- 60313/article[1]/body[1]/p[1]/collectionlink[2]: "competitive swimwear"

Fig. 5. Illustrations of the document 60313

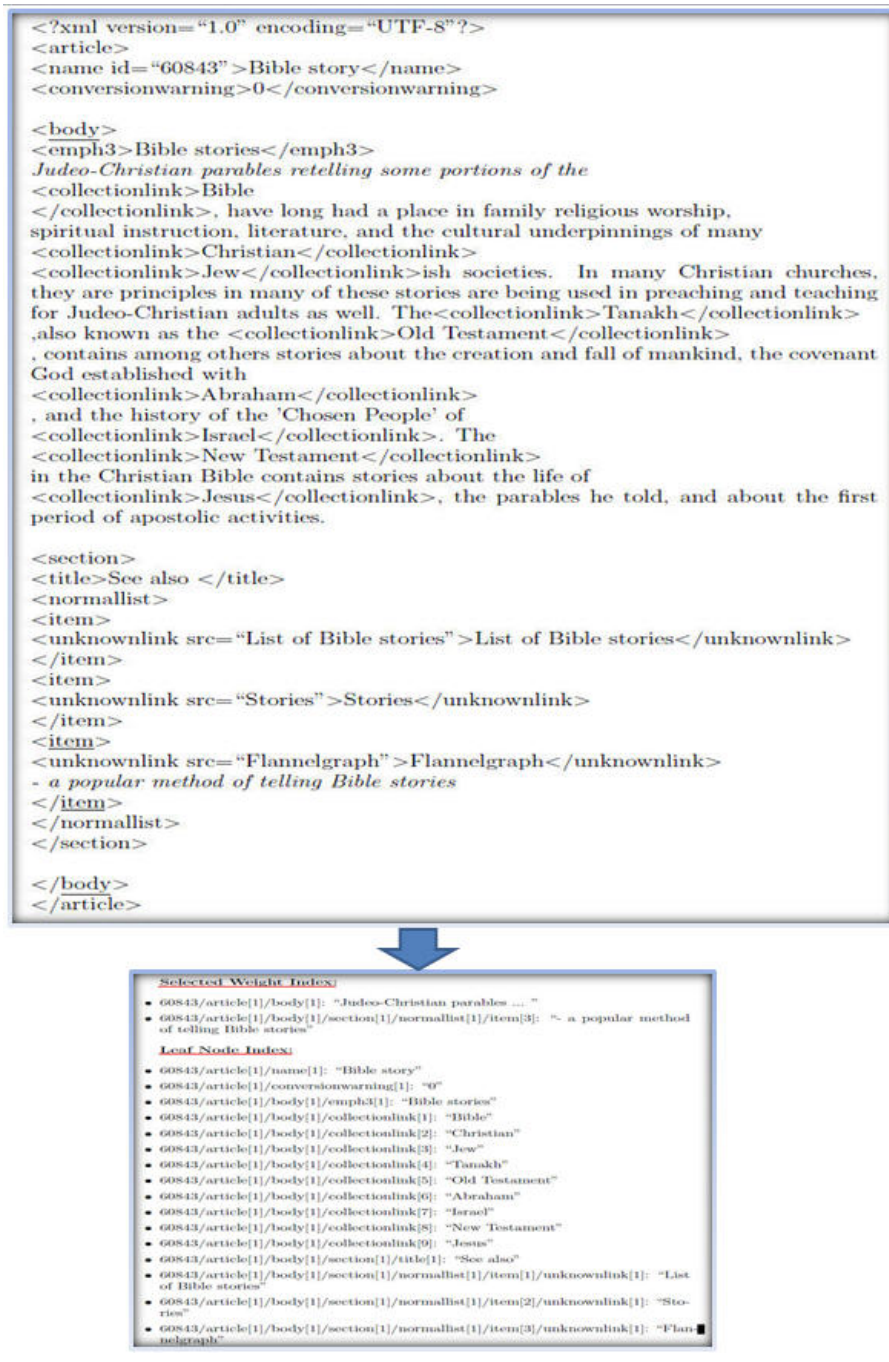

Fig. 6. Illustrations of the document 60843

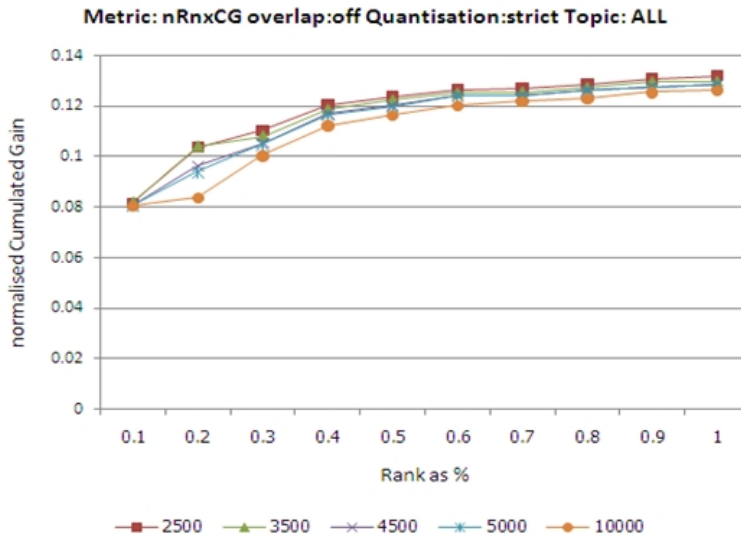

Fig. 7. The total number of leaf nodes

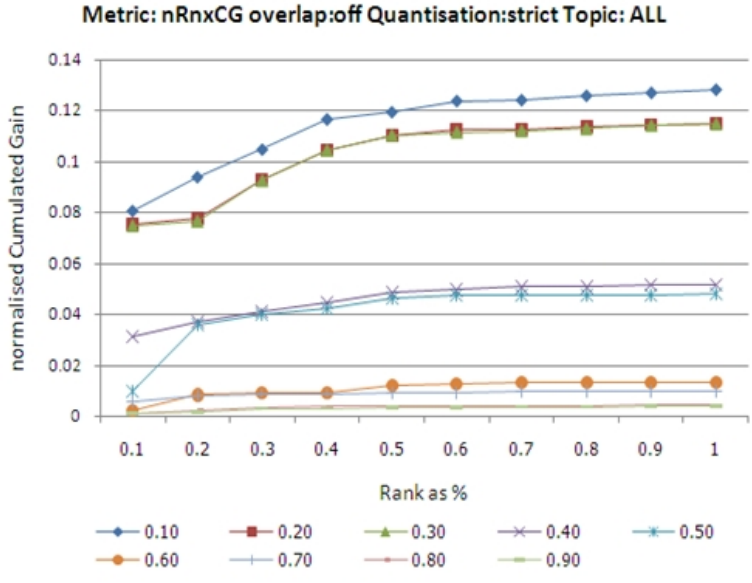

Fig. 8. Variation in the value of β parameter

in Table 4 where the columns indicate the Leaf Node index, and the rows indicate the Selected Weight index. As such, we report the effectiveness of our system for the INEX collections as follows.

MODE	Description
BOOLEAN	The final weight matches query as a *Boolean* expression
ANY	The final weight is a sum of weighted phrase ranks for matching *any* of the query words.
ALL	The final weight is the sum of weighted phrase ranks for matching *all* query words.
PHRASE	The final weight is the sum of weighted phrase ranks for matching the query *phrase*, which requires a perfect match.
EXTEND	The final weight is the sum of weighted phrase ranks and the $BM25$ weight, multiplied by a thousand and rounded to the nearest integer.

Table 4. The sphinx search modes

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.2215	0.2419	0.2386	0.2386	0.2170
ANY	0.2189	0.4419	0.4386	0.4386	0.4170
ALL	0.2350	0.4840	0.4751	0.4751	0.4768
PHRASE	0.2230	0.4595	0.4544	0.4544	0.4514
EXTEND	0.4419	0.6499	0.6499	0.6499	0.5678

Table 5. The iP[0.01] effectiveness of the INEX-2008 focused task

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.0457	0.0561	0.0560	0.0560	0.0507
ANY	0.0401	0.0961	0.0960	0.0960	0.0907
ALL	0.0521	0.0854	0.0835	0.0835	0.0829
PHRASE	0.0459	0.0870	0.0868	0.0868	0.0868
EXTEND	0.0559	0.1828	0.1827	0.1827	0.1631

Table 6. The MAiP effectiveness of the INEX-2008 focused task

The performance of different Sphinx search features can now be evaluated. Tables 5 and 6 show the results obtained from the BM25W ranking functions on INEX-2006, and Tables 7 and 8 show the results obtained from the BM25W ranking functions on INEX-2009.

Tables 5 and 6 show BM25W obtained the highest scores for INEX-2006 on 2008 topics for the MATCH EXTENDED mode on the leaf-node index and the MATCH ANY mode on the Selected Weight index, with 0.6499 for iP[0.01] and 0.1828 for MAiP, respectively. Tables 7 and 8 show BM25W obtained the highest scores for INEX-2009 on 2010 topics for the MATCH EXTENDED mode on the leaf-node

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.2011	0.2162	0.1386	0.1386	0.1170
ANY	0.2179	0.3285	0.3144	0.3284	0.2791
ALL	0.1775	0.2469	0.2432	0.2468	0.2463
PHRASE	0.1719	0.2262	0.2261	0.2261	0.2256
EXTEND	0.2198	0.3909	0.3909	0.3909	0.3769

Table 7. The iP[0.01] effectiveness of the INEX-2010 focused task

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.0205	0.0211	0.0211	0.0211	0.0199
ANY	0.0421	0.0619	0.0629	0.0624	0.0615
ALL	0.0319	0.0500	0.0535	0.0500	0.0499
PHRASE	0.0327	0.0570	0.0570	0.0570	0.0570
EXTEND	0.0455	0.0750	0.0749	0.0749	0.0728

Table 8. The MAiP effectiveness of the INEX-2010 focused task

index and the MATCH ANY mode on the Selected Weight index, with 0.3909 for iP[0.01] and 0.0750 for MAiP, respectively.

We performed a comparative study of the Sphinx search modes based on the MEXIR system. Based on the INEX evaluations on iP[0.01], the results showed that the MATCH EXTENDED mode performed better than all of the other methods for the BM25 function on the leaf-node index. Meanwhile, based on the INEX evaluations on MAiP, the MATCH ANY mode performed better than all of the other modes for term frequencies on the Selected Weight index.

In the next step, we compared BM25W with the baseline BM25 and the BM25F scoring functions. The BM25F function needed additional parameter tuning for the W_f, so we set the tuned weights for BM25F to be $W_{title} = 4.0$ and $W_{body} = 1.2$, respect to [16].

Here, we report the effectiveness of our system with respect to the INEX collections as follows.

RUN ID	**iP**[0.00]	**iP**[0.01]	**iP**[0.05]	**iP**[0.10]	MAiP
BM25	0.5451	0.5134	0.4790	0.4328	0.1476
BM25F	0.5578	0.5195	0.5014	0.4657	0.1574
BM25W	0.6579	0.6058	0.5288	0.5017	0.1728
BM25F-SS	0.5790	0.5678	0.5324	0.4938	0.1631
BM25W-SS	0.6790	0.6499	0.5724	0.5438	0.1828

Table 9. The effectiveness of the INEX-2008 focused task

The performance of different features and ranking methods can now be evaluated. Table 9 shows the results obtained from the baseline BM25, BM25F over two fields: “title” and “body” field, and BM25W using mixed content elements ranking functions on INEX-2006, while Table 10 shows the results for INEX-2009.

RUN ID	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
BM25	0.3420	0.3051	0.2090	0.1328	0.0512
BM25F	0.3477	0.3132	0.2148	0.1655	0.0540
BM25W	0.3524	0.3317	0.2315	0.1889	0.0550
BM25F-SS	0.4087	0.3732	0.2710	0.2088	0.0730
BM25W-SS	0.4131	0.3909	0.2724	0.2089	0.0750

Table 10. The effectiveness of the INEX-2010 focused task

RUN ID	INEX-2006	INEX-2009
BM25F-SS	0.5678	0.3732
BM25W-SS	0.6499	0.3909
P (t-test)	0.04	0.14

Table 11. The significance (P) is computed with a 2-tailed t-test at iP[0.01]

In order to make detailed analysis of the BM25W scoring function, we have also run experiments to study the impact of mixed content element in the performance. Table 9 shows the effectiveness of BM25W based on INEX-2006 collection. Due to the length of mixed context elements, the BM25W function exhibits an improvement of effectiveness as follows: for the baseline BM25 measured in terms of iP[0.00], iP[0.01], iP[0.05], iP[0.10], and MAiP, the improvements are 24.56 %, 26.59 %, 19.50 %, 25.65 %, and 23.85 % respectively. For the BM25F function measured in terms of iP[0.00], iP[0.01], iP[0.05], iP[0.10], and MAiP, the improvements are 17.27 %, 14.46 %, 7.51 %, 10.13 %, and 12.08 %, respectively.

Table 10 shows that the effectiveness results for BM25W are based on INEX-2009 collection; for the baseline BM25 measured in terms of iP[0.00], iP[0.01], iP[0.05], iP[0.10], and MAiP the improvements are 20.79 %, 28.12 %, 30.33 %, 57.30 %, and 46.48 %, respectively. For BM25F measured in terms of iP[0.00], iP[0.01], iP[0.05], iP[0.10], and MAiP, the improvements are 1.08 %, 4.74 %, 0.52 %, 0.05 %, and 2.74 %, respectively. The BM25W function performed well with INEX-2006, which had a relatively larger size of mixed content elements for each document (see Table 3). It can be seen that BM25W obtained the best performance, although the improvement over both the baseline BM25 and BM25F is significant for most of the considered metrics. Significance (P) was computed with a 2-tailed t-test as shown in Table 11. The BM25W improved by 0.04 % over BM25F at iP[0.01] on INEX-2006, and 0.14 % over BM25F at iP[0.01] on INEX-2009.

In this analysis, we take the results that were obtained from BM25W over mixed content elements and compare them with the results from BM25F over two fields,

RUN ID	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
GPX	0.6818	0.6344	0.5693	0.5178	0.2587
BM25W-SS	0.6790	0.6499	0.5724	0.5438	0.1828

Table 12. Comparison of the INEX-2008 focused task

viz. "title" and "body". It is shown again that BM25W works well with the mixed content elements of the document-centric XML documents. We can conclude that significant improvement of results of the BM25W function can be obtained from the mixed content elements. This finding suggests that it is possible to improve the BM25W approach, which is the usual benchmark in INEX. The main conclusion that can be drawn from the experiments is that the BM25W scoring function is successful in automatically selecting document fields by using mixed content elements, and in assigning the weight for each chosen field on query time.

In addition, the score sharing technique shows 7.27 % improvement over BM-25W, 9.29 % improvement over BM25F measured by iP[0.01] on INEX-2006, and 17.84 % improvement over BM25W, 19.15 % improvement over BM25F measured by iP[0.01] on INEX-2009.

Another conclusion which can be obtained from Table 12 is that the overall results are satisfactory, if we compare them with those usually obtained by the participants in the INEX contests. Comparing the effectiveness for the GPX system [8, 9, 10], the BM25W-SS function shows 2.44 % improvement at iP[0.01] on INEX-2006.

7 CONCLUSIONS

Generally, classical retrieval models have been dealing with flat documents without structure. The main consequence of this approach is that terms within a document are considered to have the same relevance regardless of their role in the document. This assumption implies a relevance model simplification based only on bags of words, and useful information on the structure is lost. In this paper, we first demonstrate how an automatic system can choose selected fields using mixed content elements. Our experiments confirm that the introduction of mixed content elements, and to some extent their descendants, is necessary in deriving an automatic method to choose selected fields. Secondly, we reported the experimental results of our approach using the double scoring function for retrieving large-scale XML collections using extended index schemes to handle parameter-tuned weights for each selected field. This strategy uses only common parameters, namely k_1 and b, of the baseline BM25 function. Our experiments show an improvement of up to 28 % over BM25, and up to 15 % over BM25F at iP[0.01]. In terms of processing time, our system took an average of two seconds per topic in INEX-IEEE, an average of ten seconds per topic in INEX-2006, and an average of twenty seconds per topic in INEX-2009. In addition, our approach did not require data training or an evaluation set for parameter tuning.

In future work, we plan to study the sensitivity of the evaluation to the k_1 and b parameters of BM25 for each index, and explore how to make inferences regarding structural aspects based on content and structure (CAS) queries.

Acknowledgement

We thank Miro Lehtonen for his proofreading. We also thank Prof. Jacek Kitowski and Reviewers for their intensive polishing of the paper.

REFERENCES

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

[2] Buffoni, D.—Usunier, N.—Gallinari, P.: LIP6 at INEX'09: OWPC for Ad Hoc Track. In Proceedings of the Focused Retrieval and Evaluation, and 8th International Conference on Initiative for the Evaluation of XML Retrieval (INEX'09), Shlomo Geva, Jaap Kamps, and Andrew Trotman (Eds.), Springer-Verlag, Berlin, Heidelberg 2009, pp. 59–69.

[3] Broschart, A.—Schenkel, R.: Index Tuning for Efficient Proximity-Enhanced Query Processing. In Proceedings of the Focused Retrieval and Evaluation, and 8th International Conference on Initiative for theEvaluation of XML retrieval (INEX'09), Shlomo Geva, Jaap Kamps, and Andrew Trotman (Eds.), Springer-Verlag, Berlin, Heidelberg 2009, pp. 213–217.

[4] Craswell, N.—Zaragoza, H.—Robertson, S.: Microsoft Cambridge at TREC 14: Enterprise track. In Proceedings of the TREC 14, 2005.

[5] Denoyer, L.—Gallinari, P.: The Wikipedia XML Corpus. In Proceedings of SIGIR Forum 2006, pp. 64–69.

[6] Géry, M.—Largeron, C.: UJM at INEX 2009 Ad Hoc Track. In Proceedings of the Focused Retrieval and Evaluation, and 8th International Conference on Initiative for the Evaluation of XML Retrieval (INEX'09), Shlomo Geva, Jaap Kamps, and Andrew Trotman (Eds.), Springer-Verlag, Berlin, Heidelberg 2009, pp. 88–94.

[7] Geva, S.—Murray, L. S.: XPath Inverted File for Information Retrieval. In INEX 2003 Workshop Proceedings, Germany 2004, pp. 6–13.

[8] Geva, S.: GPX – Gardens Point XML Information Retrieval INEX 2004. In: Fuhr, N., Lalmas, M., Malik, S., Szlavik, Z. (Eds.): Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML, Lecture Notes in Computer Science LNCS, Springer 2004, pp. 211–223.

[9] Geva, S.: GPX: Gardens Point XML IR at INEX 2005. In Proceedings of the 4th international Conference on Initiative for the Evaluation of XML Retrieval (INEX'05), Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai (Eds.), Springer-Verlag, Berlin, Heidelberg 2005, pp. 240–253.

[10] Geva, S.: GPX – Gardens Point XML IR at INEX 2006. In: Comparative Evaluation of XML information Retrieval Systems 5th International Workshop of the Initiative for the Evaluation Of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany 2006, Lecture Notes in Computer Science LNCS, Springer 2006, pp. 137–150.

[11] Geva, S. et al.: Overview of the INEX 2009 Ad Hoc Track. In Proceedings of the Focused retrieval and evaluation, and 8th International Conference on Initiative for the Evaluation of XML Retrieval (INEX'09), Shlomo Geva, Jaap Kamps, and Andrew Trotman (Eds.), Springer-Verlag, Berlin, Heidelberg 2009, pp. 4–25.

[12] Kamps, J.—Rijke, M. D.—Sigurbjörnsson, B.: The Importance of Length Normalization for XML Retrieval. Information Retrieval Vol. 8, 2005, No. 4, pp 631–654.

[13] Kamps, J. et al.: The University of Amsterdam at INEX 2006. In: The INEX 2006 Workshop Pre-proceedings, Schloss Dagstuhl, Germany 2006, pp. 88–99.

[14] Kamps, J.—Pehcevski, J.—Kazai, G.—Lalmas, M.—Robertson, S.: INEX 2007 Evaluation Measures. In Focused Access to XML Documents. Norbert Fuhr, Jaap Kamps, Mounia Lalmas, and Andrew Trotman (Eds.).:Lecture Notes In Computer Science, Vol. 4862, Springer-Verlag, Berlin Heidelberg 2008, pp. 24–33.

[15] Kamps, J.—Pehcevski, J.—Lalmas, M.—Robertson, S.: Encyclopaedia of Database Systems. Springer-Verlag, Heidelberg 2009, pp. 1467–1471.

[16] Itakura, K. Y.—Clarke, C. L. A.: University of Waterloo at INEX 2009: Adhoc, Book, Entity Ranking, and Link-the-Wiki Tracks. In Advances in Focused Retrieval, Seventh International Workshop of the Initiative for the Evaluation of XML Retrieval, (INEX-2009), pp. 249–259.

[17] Itakura, K. Y.—Clarke, C. L. A.: A Framework for BM25F-based XML Retrieval. In Proceedings of the 33rd International ACM SIGIR Conference on Research and development in Information Retrieval (SIGIR '10), ACM, New York, NY, USA 2010, pp. 843–844.

[18] Koolen, M.—Kaptein, R.—Kamps, J.: University of Amsterdam at INEX 2009: Ad Hoc, Book, and Entity Ranking Tracks. In Proceedings of the Focused Retrieval and Evaluation, and 8th International Conference on Initiative for the Evaluation of XML Retrieval (INEX '09), Shlomo Geva, Jaap Kamps, and Andrew Trotman (Eds.), Springer-Verlag, Berlin, Heidelberg 2009, pp. 260–272.

[19] Lalmas, M.—Tombros, A.: Evaluating XML Retrieval Effectiveness at INEX. SIGIR Forum 2007, Vol. 41, No. 1, No. 40–57.

[20] Lu, W.—Robertson, S.—MacFarlane, A.: Field-Weighted XML Retrieval Based on BM25. In Proceedings of the 4th international Conference on Initiative for the Evaluation of XML Retrieval (INEX '05), Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai (Eds.), Springer-Verlag, Berlin, Heidelberg 2005, pp. 161–171.

[21] Mass, Y. et al.: JuruXML – An XML Retrieval System at INEX '02. In: Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval 2002, pp. 73–90.

[22] Mass, Y.—Mandelbrod, M.: Component Ranking and Automatic Query Refinement for XML Retrieval. In Proceedings of the Third International Conference on Initiative for the Evaluation of XML Retrieval (INEX '04), Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltn Szlávik (Eds.), Springer-Verlag, Berlin, Heidelberg 2004, pp. 73–84.

[23] Mass, Y.—Mandelbrod, M.: Retrieving the Most Relevant XML Component. In: Proceedings of the Second Workshop of the Initiative for the Evaluation of XML Retrieval 2003, pp. 53–58.

[24] Mass, Y.—Mandelbrod, M.: Using the INEX Environment as a Test Bed for Various User Models for XML Retrieval. In Proceedings of the 4th international Conference on Initiative for the Evaluation of XML Retrieval (INEX '05), Norbert Fuhr,

Mounia Lalmas, Saadia Malik, and Gabriella Kazai (Eds.), Springer-Verlag, Berlin, Heidelberg 2005, pp. 187–195.

[25] Ogilvie, P.—Callan, J.: Languages, Models and Structured Documents Retrieval. In: Proceedings of INEX 2002, pp. 18–23.

[26] Ogilvie, P.—Callan, J.: Using Language Models for Flat Text Queries in XML Retrieval. In: Proc. of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX) 2003, pp. 97–103.

[27] Ogilvie, P.—Callan, J.: Hierarchical Language Models for XML Component Retrieval. In Proceedings of the Third International Conference on Initiative for the Evaluation of XML Retrieval (INEX '04), Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik (Eds.), Springer-Verlag, Berlin, Heidelberg 2004, pp. 224–237.

[28] Ogilvie, P.—Callan, J.: Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval. In Proceedings of the 4th International Conference on Initiative for the Evaluation of XML Retrieval (INEX '05), Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai (Eds.), Springer-Verlag, Berlin, Heidelberg 2005, pp. 211–224.

[29] Pehcevski, J.—Thom, J. A.: Hixeval: Highlighting XML Retrieval Evaluation. In Advances in XML Information Retrieval and Evaluation, The 4th International Workshop of the Initiative for the Evaluation of XML, Springer 2005, Lecture Notes in Computer Science, Dagstuhl Castle, Germany, pp. 43–57.

[30] Piwowarski, B.—Lalmas,M.: Providing Consistent and Exhaustive Relevance Assessments for XML Retrieval Evaluation. In Proceedings of the 13th ACM International Conference on Information and Knowledge Management. ACM, New York, NY, USA 2004, pp. 361–370.

[31] Ricardo, B.—Berthier, R.: Modern Information Retrieval. Addison Wesley Longman Publishing Co. Inc. 1999.

[32] Ricardo, B.—Berthier, R.: Modern Information Retrieval (Second Edition). Addison Wesley Longman Publishing Co. Inc. 2011.

[33] Robertson, S. E. et al.: Okapi at TREC-3. In D. K. Harman (Ed.), Proc. of the Third Text Retrieval Confewrence (TREC-3) 1995.

[34] Robertson, S.—Zaragoza, H.—Taylor, M.: Simple BM25 Extension to Multiple Weighted Fields. In Proceedings of the 13th ACM International Conference on Information and Knowledge Management (CIKM '04), ACM, New York, NY, USA 2004, pp. 42–49.

[35] Robertson, S.—Zaragoza, H.: The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval Vol. 3, 2009, No. 4, pp. 333–389.

[36] Rölleke, T.—Tsikrika, T.—Kazai, G.: A General Matrix Framework for Modelling Information Retrieval. Information Processing & Management, Vol. 42, 2006, No. 1, pp. 4–30.

[37] Salton, G.—McGill, M. J.: Introduction to Modern Information Retrieval. R. Donnelley Sons Company, USA 1983.

[38] Schenkel, R.—Suchanek, F. M.—Kasneci, G.: YAWN: A Semantically Annotated Wikipedia XML Corpus. In Datenbanksysteme in Business, Technologie und Web (BTW 2007), pp. 277–291.

[39] Schlieder, T.—Meuss, H: Querying and Ranking XML Documents. Journal of the American Society for Information Science and Technology Vol. 53, 2002, No. 6, pp. 489–503.

[40] Shin, D.—Jang, H.—Jin, H.: BUS: An Effective Indexing and Retrieval Scheme in Structured Documents. In: Proceedings of the Third ACM Conference on Digital Libraries 1998, pp. 235–243.

[41] Theobald, M.—Schenkel, R.—Weikum, D.: TopX and XXL at INEX 2005. In Advances in XML Information Retrieval and Evaluation: The 4th International-Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science. Dagstuhl Castle, Germany 2005, pp. 282–295.

[42] Theobald, M. et al.: TopX: Efficient and Versatile Top-k Query Processing for Semistructured Data. The VLDB Journal. Vol. 17, 2008, No. 12, pp. 81–115.

[43] Trotman, A.: Choosing Document Structure Weights. Information Processing & Management, Vol. 41, 2005, No. 2, pp. 243–264.

[44] Wichaiwong, T.—Jaruskulchai, C.: XML Retrieval More Efficient Using Double Scoring Scheme. In Proceedings of the 9th International Conference on Initiative for the Evaluation of XML Retrieval: Comparative Evaluation of Focused Retrieval (INEX '10), Shlomo Geva, Jaap Kamps, Ralf Schenkel, Andrew Trotman (Eds.), Springer-Verlag, Berlin, Heidelberg 2010, pp. 351–362.

[45] Wichaiwong, T.—Jaruskulchai, C.: MEXIR: An Implementation of High Performance and High Precision XML Information Retrieval. Computer Technology and Application, David Publishing Company, Vol. 2, 2011, No. 4, pp. 301–310.

[46] Wichaiwong, T.—Jaruskulchai, C.: XML Retrieval More Efficient Using ADXPI Indexing Scheme. The 4th International Symposium on Mining and Web, Biopolis, Singapore 2011, pp. 638–643.

[47] Wichaiwong, T.—Jaruskulchai, C.: A Comparative Study Weighting Schemes for Double Scoring Technique. The World Congress on Engineering and Computer Science, San Francisco, USA 2011, pp. 443–447.

[48] Wichaiwong, T.—Jaruskulchai, C.: A Score Sharing Method for XML Element Retrieval. Information – An International Interdisciplinary Journal, Vol. 15, 2012.

Tanakorn Wichaiwong received his B. BA. degree in business computing from North Eastern University, Khonkean in 2001, and his Master's degree in computer science from Kasetsart University, Bangkok, in 2008. He is currently working towards his Ph.D. degree in Kasetsart University, Bangkok. His current research interests include information retrieval, XML retrieval and XML query language and native XML database.

Chuleerat Jaruskulchai received her Bachelor of Education from Chulalongkorn University, and her Master's degree in applied science with specialisation in computer science from the National Institute of Development Administration in 1978. She received a Ph. D. from George Washington University, Washington, D. C., in 1998. Currently, she serves as the Chair of the Ph. D. program in computer science at the Department of Computer Science, Kasetsart University, Bangkok.