P-RANK: Efficient Ranked Keyword Search Using P-tree

Fei Pan, Imad Rahal, Yue Cui, William Perrizo
Computer Science Department
North Dakota State University
Fargo, ND 58105
Tel: (701) 231-6257
Fax: (701) 231-8255
{fei.pan, imad.rahal, yue.cui, william.perrizo}@ndsu.nodak.edu

Abstract

Nowadays, XML is becoming the standard for electronic information representation and exchange in our lives. Access to information presented in hyperlinked XML documents and other formats have always been in demand by users. In this paper, we describe the architecture, implementation, and evaluation of the P-RANK system built to address the requirement for efficient ranked keyword search over hyperlinked XML documents. Our contributions include presenting a new efficient keyword search system using a genuine data structure called the P-tree, a novel ranking method based on dimension rank voting, and a fast rank sorting method using the EIN-ring.

Keywords: Middleware, XML, P-trees, EIN-ring.

1. Introduction

The eXtensible Markup Language (XML) is a simple, very flexible hierarchical text format derived from SGML [6] to represent documents containing structured information. An XML document is made up of nested tags where each tag can also be nested. Usually, there is a single root tag for every document. Tags can have attributes, corresponding values and closing tags. Though it was originally designed to meet the challenges of large-scale electronic publishing, XML is currently also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. In fact, it is slowly becoming the de facto standard for data representation and exchange.

Access to information presented in hyperlinked XML documents has always been in demand by users. Users need to retrieve a concise ranked set of documents of interest without having to manually go through all the documents in a given collection. Users submit keywords that summarize their interests to some search utility which, in turn, returns a ranked list of document tags pertaining to the users' interests. P-RANK is a system based on this methodology and is used over both XML document collections as well as other Html or text collections. It accepts user interests in the form of keywords, matches those keywords against documents represented using P-trees, and returns a ranked list to the user.

In this paper, we describe the architecture, implementation, and evaluation of the P-RANK system built to address the requirement for efficient ranked keyword search over hyperlinked XML documents. Html and other text documents can be represented the same way by viewing them as single-tag XML documents. Our contributions include presenting a new efficient keyword search system using a data structure called the P-tree, a novel ranking method based on dimension rank voting, and a fast rank sorting method using the EIN-ring. Experimental comparisons with other ranked keyword searching approaches have shown that P-RANK is a very fast ranked keyword search system.

This paper is organized as follows. In section 2, we describe the data model and representation used in P-RANK. In section 3, we present an overview of P-RANK's system architecture. In section 4, we present the voting-based ranking method and the rank sorting method using the EIN-ring. We experimentally compare our approach to the traditional inverted list approach in section 5. Finally, we conclude this paper in section 6.

2. Data Model and Representation

In this section, we first present the vector model data representation, and then briefly review a novel compressed column-wise quadrant-based tree structure, the P-tree [1].

2.1 Data Representation

The vector space model is a simple ubiquitous approach used to represent text documents in machine understandable formats [3] [4] [5]. It is characterized by being relatively computationally efficient and by having conceptual simplicity; however, it suffers from loss of information related to the original document structure such as the order of the terms relative to each other or the frontiers between sentences and paragraphs–document tags in the case of XML. Each document is represented as a vector whose dimensions are all the terms in the existing document collection; the set of terms used as dimensions is referred to as the term space. In this representation, vector coordinates are terms with numeric values representing their relevance to the corresponding documents where higher values mean higher relevance. This process of giving numeric values to term coordinates is referred to as term weighting in the literature. We can view weighting as the process of giving more emphasis to more important terms.

Term weight measures in document vectors can be binary with the values 1 and 0 denoting the existence or absence of the term in the corresponding document, respectively.

Comparisons over this binary representation are usually fast and efficient. However, due to the great loss of information associated with it, this representation lacks the accuracy needed. For example, a term *t* might exist 100 times in document *d1* and only 1 time in another document *d2* of similar size in which case it is clear that *t* is more related to *d1*'s context than to *d2's*; however, using the binary representation, this information is lost.

A more sophisticated representation records the exact frequency measure of each term in a document. The simple term frequency (TF) records the number of occurrences of each term *t* in the document. As the TF measure increases, *t* becomes more related to the document context. The inverse document frequency (IDF) records the ratio between the total number of documents and the number of documents in which a certain term, *t*, exists. It is calculated as: IDF(*t*) = Log2(N/N*t*), where *t* is the term under consideration, N is the total number documents in our collection and N*t* is the number of documents containing at least one occurrence of *t*. Notice that the IDF(*t*) value increases as N*t* decreases and thus as the term becomes more unique. This is due to the fact that globally unique terms–i.e. terms that don't exist in many documents–have greater potential in identifying specific documents when specified in the user's search criteria.

From the last two frequencies, a third type of frequency, the weighted term frequency (WTF), is derived. WTF is calculated as the product of the above two frequencies, WTF = TF x IDF(*t*). It should be clear that the WTF is more accurate than the all the representations presented thus far as it combines the advantages of the TF and IDF.

P-RANK uses the ubiquitous vector space model over XML documents. The WTF is used for dimension values in term vectors; however, P-RANK applies the WTF to each document tag rather than to the whole document. This is due to the fact that XML search engines usually return document tags and not documents as opposed to Html and other text search engines. Notice that by doing so, we can alleviate one of the aforementioned problems associated with the vector space model, namely, the loss of information related to model's inability to express paragraphs frontiers. The severity of this problem is now reduced because each tag is represented separately as a term vector. We will refer to the structure resulting from representing document tags in term vector format as the document tag by term matrix.

In addition to using all term weights as dimensions in the document vector, two other weights are employed by P-RANK: document reference weight and tag depth weight. The main motivation for those two weights is to highlight the characteristics pertaining to hyperlinked XML documents as opposed to other Html or simple text documents in order to include them in the ranking evaluation. The document reference weight is an integer value given to each document, *d*, to express the referencing importance of *d* to other documents in the document set. It is calculated as the total number of document references (IDREFs and Xlinks [8]) to *d*. All the tags of a document will have the same reference weight since references are usually document based and not tag based.

The depth weight reflects the hierarchical structure of an XML document and is calculated per document tag. The deeper a tag's position is in a document, the more specific its contents are and thus they should be weighted higher [8]. P-RANK uses a simple incremental weight value to accomplish this purpose. The root tag of a document is weighted 1, first level tags are weighted 2, second level tags are weighted 3, and so on until we reach the lowest level tags which get the highest weights depending on their depth.

2.2 Structure of P-trees

As noted earlier, P-RANK is built on a novel and genuine data structure, the P-tree (Predicate tree). P-trees are quadrant-based tree structures that store numeric relational data in a loss-less bit-compressed column-wise format by splitting each attribute into bits, grouping bits in each bit position, and representing each bit group by a P-tree [1]. A P-tree can be 1-dimensional, 2-dimensional, 3-dimensional, etc. In this brief review, we will focus on 1-dimensional P-trees.

Given a data set with d attributes, $X = (A_1, A_2 \ldots A_d)$, and the binary representation of j^{th} attribute, A_j, as $b_{j,m}b_{j,m-1}...b_{j,i}\ldots b_{j,1}b_{j,0}$, we decompose each attribute into bit groups, one group for each bit position. To build a P-tree, a bit group is recursively partitioned into halves and each half into sub-halves until the sub-half is pure (i.e. entirely 1-bits or entirely 0-bits).

The detailed construction of P-trees is illustrated by an example in Figure 1. We represent the attribute value in binary format, e.g., $(7)_{10} = (111)_2$, as shown in a). Then, we vertically decompose the attribute value into three separate bit groups, one group for each bit position, as shown in b). The corresponding basic P-trees, P_1, P_2 and P_3, are constructed by recursive partitioning, which are shown in c), d) and e). Each P-tree will give the total number of 1s in the bit group on the root level. The first node from the left on the second level will give the number of 1s in the first half of the bit group, and, similarly, the second node will give the number of 1s in the second half of the bit group. This logic is continued throughout the tree with each node giving number of 1s in either the first or the second half (depending on whether it is the left or right node) of the bit group represented by the parent node. As shown in c) of Figure 1, the root of P_1 tree is 3, which is the 1-bit count of the entire bit group. The second level of P_1 contains the 1-bit counts of the two halves, 0 and 3. Since the first half is pure, there is no need to partition it further. The second half is further partitioned recursively.

AND, OR and NOT logic operations are the most frequently used P-tree operations. For efficient implementation, we use a variation of P-trees, called Pure-1 trees (P1-trees). A tree is pure-1 if all the values in its sub-trees are 1's. A node in a P1-tree is a 1-bit if and only if the corresponding half it represents is pure-1. Figure 2 shows the P1-trees corresponding to the P-trees in c), d), and e) of Figure 1.

The P-tree logic operations are performed level-by-level starting from the root level. They are commutative and distributive, since they are simply pruned bit-by-bit operations. For instance, ANDing a pure-0 node with anything results in a pure-0 node, ORing a pure-1 node with anything results in a pure-1 node. In Figure 3, a) is the ANDing result of $P1_1$ and $P1_2$, b) is the ORing result of $P1_1$ and $P1_3$, and c) is the result of NOT $P1_3$ (or $P1_3$'), where $P1_1$, $P1_2$ and $P1_3$ are shown in Figure 2.

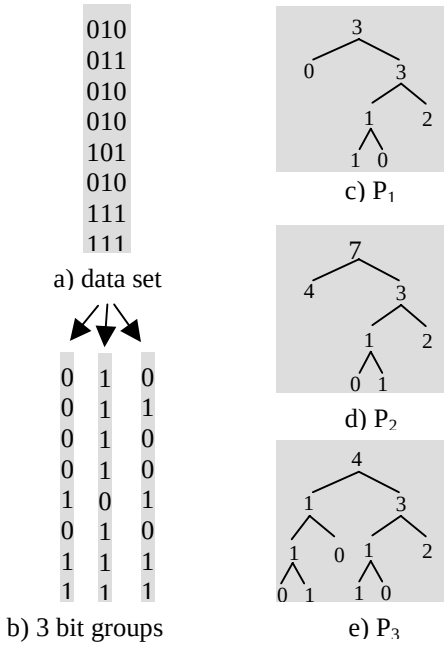

Figure 1. Construction of 1-D Basic P-trees

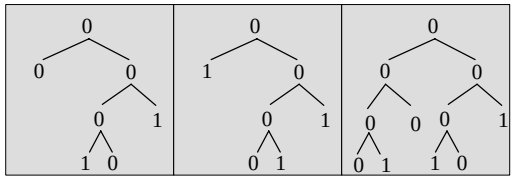

Figure 2. P1-trees for the transaction set

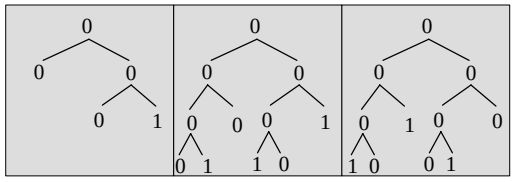

Figure 3. AND, OR and NOT Operations

3. P-RANK System Architecture

Figure 4 depicts an abstract level view of the architecture of P-RANK. Documents are transformed to a document tag by term matrix using the weighting schemes and the optional pre-processing steps discussed in [12]. This matrix is then, in turn, converted to its P-tree representation by using the P-tree Capture Interface [13]. The result of this last step is fed into the P-RANK engine. When users submit keywords to the system, P-RANK, first, applies on the keywords any optional pre-processing steps that were performed on the documents. This includes reducing keywords to their original forms or filtering some of them by using stop lists. After that, P-RANK does the required matching of the pre-processed keywords by EIN-ring-based ranking and sorting as will be discussed in Section 4. This is accomplished using the P-tree version of the document tag by term matrix. Finally, a ranked list of XML document tags matching his/her keywords is returned to the user.

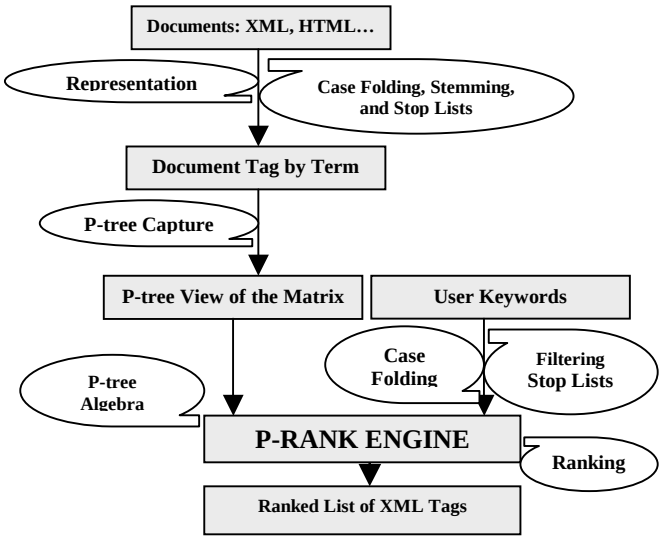

Figure 4. Architecture of P-RANK

4. Efficiently Ranking Keyword Search Queries

In this section, we propose a new ranking method based on weight voting. We will also present an efficient result sorting method using EIN-rings and P-trees.

4.1 Ranking by Votes

Traditional methods for ranking queries results, such as Google [9] [10], are based on the use of fixed ranking formulas, which integrate different dimension weights into the ranking score and evaluate documents accordingly. This scheme is straightforward and natural; however, it suffers from the weight shifting domination problem described next.

Usually, the result list of document tags that is of interest to the user is specific to the user's query. As a result, for each specific query and returned result list, the value distribution of

the normalized dimension weights is different. In the result list, some weights may have very large values compared to others. By using a fixed formula to integrate all the weights together, the large weights will dominate in the final rank score. We refer to this situation as the "shift domination" phenomenon because the "domination" of weight values will change with the query. A simple and straightforward solution in this case is to normalize the weights for each query so that they occur within some defined range and then integrate them into final rank score. However, the result will still suffer from the domination problem in the case of skewed weight distribution.

P-RANK uses a more interesting scheme based to handle the "shift domination" phenomena. Instead of using value normalization and fixed-formula integration, we propose a novel ranking method based on voting by the order of the dimension weights and not by their actual values. After finding the result list that matches a certain query, P-RANK determines the rank order of each dimension value for each of the returned document tags separately, and then derives the rank order of a tag by using a summation over the dimension rank orders.

Figure 5 shows an example of returned list with the dimension values converted to rank order values. Using a simple summation formula over the returned list will give Tag1 a rank weight of 46, Tag2 a rank weight of 27 and Tag3 a rank weight of 152; as a result, Tag3 will rank before Tag1 which will rank before Tag2. In our approach, we derive the rank order of a tag by using a summation over the dimension rank orders; so, Tag1 will weigh 7 and will be ordered first, followed by Tag2 with a weight of 6 and then Tag3 with a weight of 5.

	D1	D2	D 3
Tag 1	23	12	11
Tag 2	2	11	14
Tag 3	1	1	150

	D1	D2	D 3
Tag 1	3	3	1
Tag 2	2	2	2
Tag 3	1	1	3

Figure 5. A returned list with rank order values

It is obvious that our ranking approach will lead to different ranking results than traditional formula-based approaches that operate directly on dimension weights. This is mainly due to our approach's "democratic" dimension voting resulting from dimension ranking and our ability to alleviate the weight "shift domination" phenomena that is ubiquitous among formula-based approaches.

4.2 EIN-ring based Keyword Queries Ranking

In this section, we give a brief review of the EIN-ring followed by the issue of ranking the results of keyword search queries over XML documents.

4.2.1 EIN-ring

We describe a unique equal interval neighborhood ring (EIN-ring) [11], which is used for ranking results of keyword queries. We first define neighborhood rings and EIN-rings, and then give some propositions on the calculation of EIN-rings using P-trees. We use $\wedge$, $\vee$ and prime (') to denote the P-tree operations AND, OR and NOT, respectively.

Definition 1. *The Neighborhood Ring* of data point c with radii r_1 and r_2 is defined as the set $R(c, r_1, r_2) = \{x \in X \mid r_1 < |c-x| \leq r_2\}$, where X is the space and |c-x| is the distance between x and c.

Definition 2. *The Equal Interval Neighborhood Ring* of data point c with radius r and fixed interval value λ is defined as the neighborhood ring $R(c, r, r+\lambda) = \{x \in X \mid r < |c-x| \leq r+\lambda\}$, where X is the space and |c-x| is the distance between x and c. $r = k\lambda$ for k=1, 2,…, and the corresponding rings are called the *k^{th} EIN-rings*. Figure 6 shows 2-D EIN-rings for k = 1, 2, and 3.

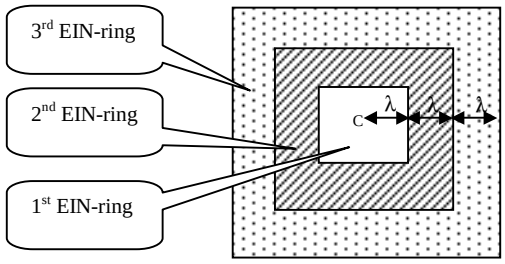

Figure 6. Diagram of EIN-rings.

The interval λ is a user-defined parameter based on accuracy requirements. The higher the accuracy requirement, the smaller the value of λ is. The calculation of EIN-rings is accomplished using a range predicate tree. A range predicate tree, $P_{x \prec y}$, is a basic P-tree that satisfies predicate $_{x \prec y}$, where x is a some data in a data set X, y is a boundary value, and $\prec$ is the comparison operator such as <, >, ≥, or ≤. The calculation of a range P-tree, $P_{x \prec y,}$ is given as follows.

Let x be an (m+1)-bit data value in a data set X, and P_m, $P_{m-1,}$ …, P_0 be the P-trees for the vertical bit groups of X as described in Section 2.2. Let $v=b_m \ldots b_i \ldots b_0$, where b_i is the value of the i^{th} binary bit in v, and $P_{x \geq v}$ be the predicate tree for the predicate x≥v. $P_{x \geq v} = P_m \, op_m \, \ldots \, P_i \, op_i \, P_{i-1} \, \ldots \, op_1 \, P_0$, where i = 0, 1 … m, op_i is $\wedge$ if b_i=1 or $\vee$ otherwise, and the operators are right binding. As aforementioned, $\wedge$ denotes the AND operation and $\vee$ denotes the OR operation. Right binding states that operators are associated from right to left; e.g., $P_2 \, op_2 \, P_1 \, op_1 \, P_0$ is equivalent to $(P_2 \, op_2 \, (P_1 \, op_1 \, P_0))$. An example of the calculation of $P_{x \geq v}$ is $P_{x \geq 101.} = (P2 \wedge (P_1 \vee P_0))$.

Calculation of $P_{x \leq v}$ is very similar to the calculation of $P_{x \geq v}$. : Let x be an (m+1)-bit data value in a data set X, and P'_m, P'_{m-1}, … P'_0 be the complement P-trees for the vertical bit groups of X (i.e., each P'_i is the binary complement of P_i). Let $v=b_m \ldots b_i \ldots b_0$, where b_i is value for the i^{th} binary bit of v, and $P_{x \leq v}$ be the predicate tree for the predicate x≤v. $P_{x \leq v} = P'_m op_m \, \ldots$

$P'_i\ op_i\ P'_{i-1}\ \ldots\ op_{k+1}P'_k$, where $k \le i \le m$, op_i is $\wedge$ if $b_i=0$ or $\vee$ otherwise, k is the rightmost bit position with value of "0" (i.e., $b_k=0$ and $b_j=1, \forall\ j<k$), and the operators are right binding. An example of the calculation of $P_{x \le v}$ is $P_{x \le 101} = (P'_2 \vee P'_1)$.

4.2.2 Ranked Sorting Using EIN-rings

We now present the ranked sorting for keyword queries over XML documents using EIN-rings. Consider a keyword search query *Q = (k1, k2, …, kn)*, where *k1, k2, …,kn* are the input keywords. Let P_{mask} be the P-tree representing the result list that contains all the XML tags containing all the input keywords. A simplified prototype vector model as discussed in Section 2 is shown in Table 1. We encode each column in Table 1 by bits and represent each bit position by a P-tree. The calculation of P_{mask} is as follows: First, we find the corresponding input keywords in the Term Existence section of Table 1; the P-tree representing the result list of the tags containing all the input keywords, P_{mask}, is just the "ANDing" of all P-trees for participating keywords, that is $P_{mask}=P_{k1} \wedge P_{k2} \ldots \wedge P_{kn}$ where P_{k1}, P_{k2}, …, and P_{kn} are the P-trees for the participating keywords.

Table 1. Simplified Prototype of the Data Model

	Term Existence			**Term Weight: W1**			**Ref. Weight W2**	**Depth Weight W3**
	T1	…	Tk	T1	…	Tk		
Tag_1	1/0	1/0	1/0	$W1_{11}$	…	W1k1	$W2_1$	$W3_1$
Tag_2	1/0	1/0	1/0	$W1_{12}$	…	W1k2	$W2_2$	$W3_2$
…				…	…	…	…	…
Tag_n	1/0	1/0	1/0	$W1_{1n}$	…	W1kn	$W2_n$	$W3_n$

Consider each weight in W1, W2 and W3 sections in Table 1 as a dimension of the search space X. We have a space of (k+2)-dimensions (W1 having k term dimensions, W2 and W3, so, k+2 in total), $X = (W_{11},\ldots, W_{1k}, W_2, W_3)$, and we denote the set of EIN-rings by $R(X_{start}, r, r+\lambda)$, where $X_{start} = (x_{start\text{-}1}, x_{start\text{-}2}, .., x_{start\text{-}k+2})$ is the center for the rings of radii r, $r = r_1, r_2, \ldots r_{k+2}$, where each r_i is a radius associated with dimension W_i and λ is a fixed interval. Each r_i is referred to as the internal radius of the ring with center X_{start} and $r_i+\lambda$ is the corresponding external radius of the ring. The calculation of points within EIN-rings $R(X_{start}, r, r+\lambda)$ is presented as follows.

X_{start} is the point in X having the largest weight values, i.e., $x_{start\text{-}i} = \max(W_i)\ \forall$ i. Initially, each r_i is zero; then, r_i is increased in steps each equal to λ until we reach the $\max(W_i)$. Each step yields a bigger EIN-ring encompassing all the rings generated before it. Let $P_{ri,\lambda}$ be the P-tree representing all points Y in X existing within the scope of the EIN-ring $R(X_{start}, r_i, r_i+\lambda)$. Note that $P_{ri,\lambda}$ is just the predicate tree corresponding to the predicate $X_{start\text{-}i} - r_i - \lambda < Y \le X_{start\text{-}i} - r_i$ where Y is the set of points $(w_1, w_2, \ldots, w_{k+2})$ in X. We calculate $P_{ri,\lambda}$ by ANDing $P_{Y<Xstart\text{-}i\text{-}ri}$ and $P'_{Y<Xstart\text{-}i\text{-}ri\text{-}\lambda}$. For the sake of clarity, we will refer to $P_{Y<Xstart\text{-}i\text{-}ri}$ and $P'_{Y<Xstart\text{-}i\text{-}ri\text{-}\lambda}$ as P_{min} and P'_{max} respectively. P_{min} is shown in the shadow area of a) and P'_{max} in the shadow area of b) in Figure 7. According to the calculation formula of EIN-ring, P_{min} is calculated as

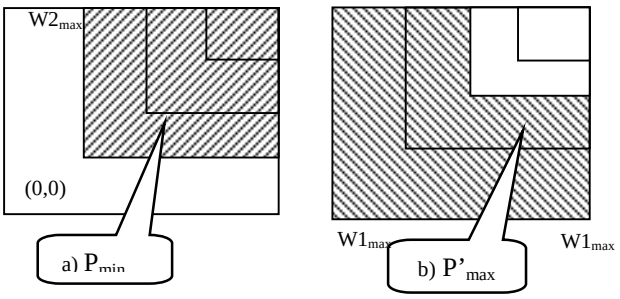

Figure 7. Calculation of Data Points within EIN-ring $R(X_{start}, r, r+\lambda)$.

$$P_{min} = P_{Y<Xstart\text{-}i\text{-}ri} = P_{w1 \le Xstart\text{-}1\text{-}r1} \wedge P_{w2 \le Xstart\text{-}2\text{-}r2} \wedge \ldots \wedge P_{wk+2 \le Xstart\text{-}k+2\text{-}rk+2}, \quad (1)$$

where each $P_{wi \le Xstart\text{-}i\text{-}ri}$ is calculated as explained previously in this section. Similarly we calculate P_{max} as

$$P_{max} = P_{X<Xstart\text{-}i\text{-}ri\text{-}\lambda} = P_{w1 \le Xstart\text{-}1\text{-}r1\text{-}\lambda} \wedge P_{w2 \le Xstart\text{-}2\text{-}r2\text{-}\lambda} \wedge \ldots \wedge P_{wk+2 \le Xstart\text{-}k+2\text{-}rk+2\text{-}\lambda}, \quad (2)$$

P'_{max} is the complement of P_{max}. $P_{ri,\lambda}$ can be calculated by ANDing P_{min} and P'_{max}. Note that the set of points in $P_{ri,\lambda}$ that contain all the keywords can be derived by ANDing the P_{mask} with $P_{ri,\lambda}$.

The set of EIN-rings along every dimension gives us an intrinsic ranking for the points along that dimension. Points existing in the same ring have the same rank value while points existing in other rings have different rank values. Rings with smaller external radii get higher rank values along that dimension. For example, points in the ring with a radius of $\max(W_i)$ get a rank value of 1, points in the ring with a radius of $\max(W_i)-\lambda$ get a rank value of 2, points in the ring with a radius of $\max(W_i)-2\lambda$ get a rank value of 3 and so on and so forth. After obtaining all the rank values using the EIN-rings for all points along every dimension, we calculate the rank order of the points in space as the weighted summation of all dimensions rank values. The weights for the dimensions can be derived by domain experts or machine learning algorithms. The detailed discussion of dimension weights is beyond the scope of this paper.

To accomplish our task, we exploit a fast binary search strategy. We calculate all the tags in the space by dividing the space into halves and traversing each half recursively until we cover all the space. The pseudo-code of the algorithm for finding all tags within successive rings using binary search strategy is shown in Figure 8.

Algorithm: EBS(&Wstart, &Wend)
Input: Start point and End point in d-dimension weight space X
Output: Set of tags within successive rings

```
1. for i=0 to d-1
2. if Wend[i]<Wstart[i]+λ
3. mid[i] = ⌊( Wend[i]+ Wstart[i])/2⌋
4. sh[i]=EBS(Wstart[i],mid[i])
5. sl[i]=EBS(mid[i],Wend[i])
6. return Set=(sl[i],sh[i])
7. else
8. return Set ⊆ Pr,λ[i]∧Pmask
9. endif
```

Figure 8. Algorithm for finding tags within successive rings

5. Performance Analysis

An implementation for P-RANK has been developed in C++ and executed on a 1 GHz Pentium PC machine running Debian Linux 4.0 with 1GB main memories. We compare P-RANK with the ranking approach based on inverted lists. The inverted list (IL) is widely used in keyword search methods for both hyperlinked Html and XML documents. We implemented two naïve IL-based versions, one was ordered by the ID approach (IL-ID) and another was ordered by rank approach (IL-rank). The P-RANK implementation adhered to the algorithmic description presented herein. Details for the P-tree implementation can be found in [11]. We experimented on the DBLP data set [13]. Performance results using randomly generated keywords are shown in Figure 9.

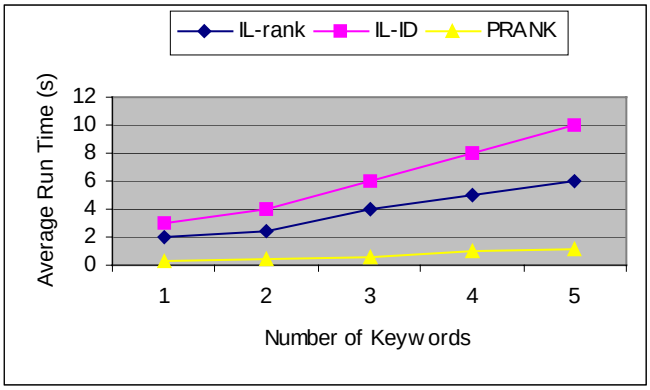

Figure 9. Comparison of IL-rank, IL-ID, and P-RANK.

As shown in Figure 9, P-RANK is orders of magnitude faster than IL-ID and IL-rank on single and multiple keyword-ranking searches, which indicates the superiority of P-RANK's keyword ranking speed.

6. Conclusion

In this paper, we described the architecture, implementation, and evaluation of the P-RANK system built to address the requirement for efficient ranked keyword search over hyperlinked XML documents. P-RANK's contributes by presenting a new and efficient keyword-based search system characterized by speed and compression due to the use of P-trees. P-RANK's ranking is based on the novel dimensional rank order weighting method; in addition, its sorting method is very fast due to the use of EIN-rings. Experimental comparisons with other approaches demonstrated P-RANK's superiority of keyword ranking speed.

References

[1] Perrizo, W., *Peano count tree technology lab notes*. Technical Report NDSU-CS-TR-01-1, 2001. http://www.cs.ndsu.nodak.edu/~perrizo/classes/785/pct.html. January 2003.

[2] M. Khan, Q. Ding, and W. Perrizo, *K-nearest neighbor classification on spatial data streams Using P-trees*. Proceedings of the PAKDD, Pacific-Asia Conference on Knowledge Discovery and Data Mining (Taipei, Taiwan), 2002.

[3] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval." Information Processing & Management, 24(5), 513-523, May 1988.

[4] G. Salton and M.J. McGill, *Introduction to modern information retrieval*. McGraw-Hill, New York, 1983.

[5] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing." Communications of the ACM 18(11), 613--620, November 1975.

[6] International Organization for Standardization (ISO 8879, 1986) http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387. July 2003.

[7] World Wide Web Consortium. http://www.w3.org.

[8] L. Guo, F. Shao, G. Botev, and J. Shanmugasundaram, *XRANK: Ranked keyword search over XML documents*. Proceedings of the ACM SIGMOD, Special Interest Group in Management of Data (San Diego, California), 16-27, 2003

[9] S. Brin, L. Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, WWW Conf., 1998.

[10] J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment", JACM 46(5), 1999.

[11] F. Pan, B. Wang, B., and W. Perrizo, *Rapid, Accurate, Scalable Nearest-Neighbor Classification*. http://www.cs.ndsu.nodak.edu/~datasurg/papers/review/03_ENN.pdf. July 2003.

[12] I. Rahal and W. Perrizo, *An optimized P-tree based KNN approach for Text Categorization*. http://www.cs.ndsu.nodak.edu/~rahal/research/textmining/ICDM_R214.pdf. July 2003.

[13] P-tree World Wide Web Homepage. http://midas-11.cs.ndsu.nodak.edu/~ptree/

[14] XML Data Repository contains DBLP data set. http://www.cs.washington.edu/research/xmldatasets/www/repository.html. July 2003.