RESEARCH ARTICLE OPEN ACCESS

Efficient Query Processing For PIM System

U. Padma Jyothi[1], A.Seenu[2]
Department of Computer Science and Engineering
SVECW, Bhimavaram, West Godavari District
Andhra Pradesh - India

ABSTRACT
The amount of personal data which is stored in the personal information management is increasing rapidly. There is a need of powerful search tools to often access the data efficiently. Most of the users use hierarchical directory structure to organize their files. Users may not know the exact location and other details about the data for what they are searching. The Search operation should allow for some approximations during query processing to obtain the results. A novel multidimensional fuzzy search approach was proposed that allows users to perform approximate search across three dimensions such as structure, metadata and content. We present a scoring framework for content, metadata and structure and they are unified into single dimension. In this proposed system we make use of indexes and algorithms to identify the relevant file, and improve the accuracy.
Keywords:- Multidimensional search, personal information management system, query processing, Scoring framework.

I. INTRODUCTION

The data which is to be stored in personal information management systems (PIM)[1] is rapidly increasing, Personal information management system can be defined as the activity in order to organize, maintain and access the data which is available in personal information items such as files, images, emails and music. Where the personal information management system is one of the application of dataspace- is an abstraction method which is mainly used to overcome some of the problems of data integration systems. The dropping price of per byte storage in personal system may need a powerful search tools to access often very disparate data in a simple and efficient manner. Such type of tools should provide both high quality scoring mechanisms and efficient query processing capabilities.

In order to perform keyword search and locate personal information stored in file there are numerous search tools such as the commercial tools Google Desktop Search [2] and Spotlight [3]. These tools are discontinued because of some problems such as slow, buggy, and also difficult to maintain. These commercial tools mainly concentrate on textual part of the query for searching a file—similar to what has been done in the Information Retrieval (IR) community, but this community consider structure (file path) and metadata (e.g.,file type, title, author) as filtering conditions.

Recently, the research community has turned its focus on search over to Personal Information management and Dataspaces [4,5,6], which consist of disparate data collections. However, in these search tools, the work is mainly focused on Information Retrieval style keyword queries and use other system information to guide the keyword-based search.

Keyword based search is often insufficient, for example if a user want to search for a particular file, the search is successful if and only if the exact information is provided in a query. But by using this novel approach we can perform the successful search, even the given data is approximate.

For example: Consider an administrator of a college saving personal information (college data) in the file system of a personal computing device. The admin wants to retrieve a photo of a passed out student because of some reasons. Unfortunately, admin does not remember that the student belongs to which year and branch. The file directory structure would have been created and maintained consistently and all photos properly indexed. In real life scenarios, this is rarely the case: users change their file organizations over time, inconsistently gloss their data and may gather information from multiple places. In our example, administrator has changed the way he organizes his photos over time when he switched to a new computer and decided to use a new photo organization software, and his pictures are not consistently indexed. As a result, photos of students in distinct years match different directory structures and do not necessarily have matching tags, as shown in Figure 1.

This directory structure may complicates the search for specific pictures. A content search consider only the term “Vishnu”, where it is likely to obtain matches, of various relevance. None of the pictures which are mentioned in the figure1 are of exact match. There is some approximation in the structure or the metadata of the directory structure of personal information management.

Another possibility is to write a a query that takes into account the directory structure as well as the structural metadata information of a file.

```
[Filetype=*.png AND
Content="Vishnu" AND
Structure=/photos/aa/batch14]
```

Current tools would answer this query by returning all files of type **.png* under the directory */photos/aa/batch14* (filtering conditions) that have content similar to "Vishnu". Because the directory structure and the type of file are used as filtering conditions, only the exact match Vishnu.png would be returned as a result to this query, files that are very relevant to the content search part of the query, but which do not satisfy these exact conditions are not considered as valid answers. For example, consider a file 0545.png, which contains the same extension and also same path but is not in the query target directory would not be returned although it may be a suitable approximate match to the query.

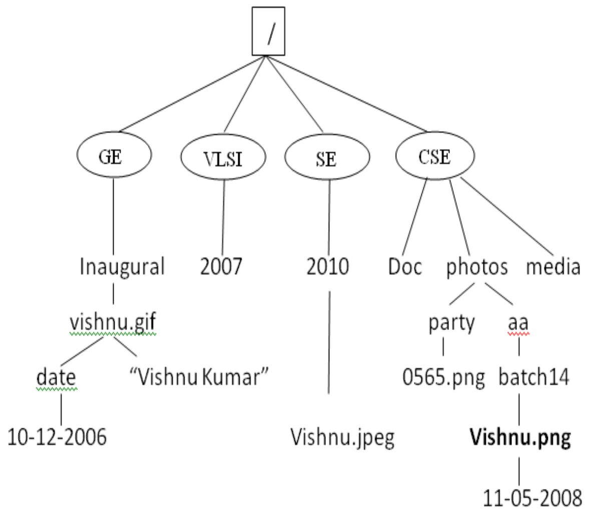

Figure 1: A Subset of file structure

Because of the nature of personal information systems, we believe it is critical issue to support approximate matches on the content and structural components of the query. In previous work [7, 8], we presented a scoring framework that considers relaxed query conditions on several query dimensions. Our approach individually scores each dimensions such as the content, metadata, and structure dimensions and combines the resulting scores in a unified scoring framework using an *IDF*-based scoring for each dimension. We proposed an efficient and dynamic index structures to support our scoring techniques across three dimensions.

In this paper, we extend our query model to consider structure within the file and relax structural conditions across the file boundaries. Data and query model are described in section 3.

II. SYSTEM DESIGN

The system design is the process in which the service provider can perform the search operation based on dimensions such as content, metadata and structure from the repository i.e, personal information based on the DAG_path algorithm is used to retrieve the path of the file for a given query. Finally the results are obtained based on the given query.

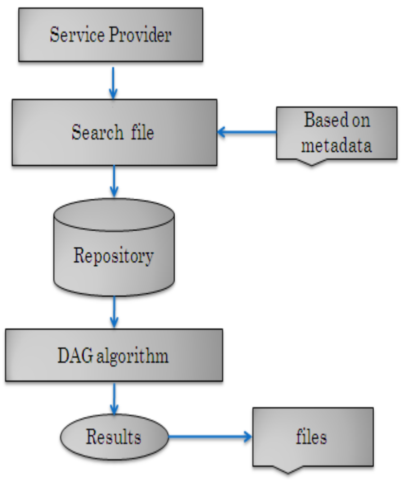

Figure 2: System Architecture

III. UNIFIED DATA AND QUERY MODEL

The personal information data model (Figure 1) mainly considers structure both direct files and the files stored in a directory in a combined manner. The complete directory structure can be seen as an XML document, the leaf nodes consists of file, and internal nodes represents more general files i.e, the union of the all the leaf nodes. This novel multidimensional fuzzy search currently supports relaxed query conditions on three dimensions: *content* for conditions on the files (text content), *metadata* for conditions on the system information about files (creation time, date), and *structure* for conditions on both the directory paths to access files and the internal structure of the file.

In this paper, lets us consider structure dimension. Our relaxed queries can be viewed as XQuery [9] expressions, it is a query language designed to retrieve structured and unstructured data. Consider physical files as query results. Which allows for various levels of granularity of query results. A file which matches with a particular query has potential to answer a query and such type of files gets assigned a score for each dimension based on how close it matches to the particular query condition.

Consider metadata dimension, we introduce a hierarchical relaxation approach for each type of searchable metadata to assign score. A main characteristic is hierarchical representation is critical; that is the set of files matching a nodes should be equal or include the set of files matching each of its children nodes.

Scores across multiple dimensions are unified into a single framework so that the score is used for ranking of answers. Our scoring strategy was mainly based on an *IDF*-based interpretation of scores, as introduced in [10]. For each of the query condition, The score for files are obtained based on the *least relaxed* form of the query that each file matches. Scoring all the dimensions are mainly based on a *IDF*-based (Inverse Document Frequency) which permits us to meaningfully aggregate multiple single dimensional scores into a unified multidimensional score. Our structure scoring strategy extends prior work on XML structural query relaxations [11]. In particular, we use several types of structural relaxations, to handle the needs of user searches in a file system.

Assuming that structure query conditions are given as structures, these relaxations are:

•**Edge Generalization** is used to relax a parent-child relationship to an ancestor-descendant relationship. For example, applying generalization of edge to */photos/batch14* would result in */photos//batch14*.

• **Path Extension** is used to extend a path *P* such that all files within the directory sub tree rooted at *P* are treated as results. For example, applying extension of path to */photos/aa* would result in */photos/aa//*∗ .

• **Node Deletion** is used to drop a node from a path. For example, applying deletion of node on *aa* from */photos/aa /batch14* would result in */photos//batch14*.

• **Node Inversion** is used to permute nodes within a path. For example, applying inversion of node on *aa* and *batch14* from */photos/aa/batch14* would result in */photos/*(*aa/batch14*),allowing for both the original query condition as well as */photos/batch14/aa*.

• **Node Extension** is used to allow for structural conditions to be matched by the content information contained within the files. For example, applying node extension on */photos/aa* would result in the query */photos/{aa}*, whose meaning is that the term “aa” can be either external directory structure or internal file or file content.

We can say that a file *matches* a particular query condition if all structural relationships between the condition’s components are preserved in the file’s unified external and internal structure.

Finally, given a directory structure *d* and query *q*, the structure score of a file *f* with respect to *q* is computed as:

$$Score_{structure} = \max_{x \in Y(q)} \{ score_{idf}(x) \mid f \in F(d,x) \} \quad (1)$$

$$Score_{IDF}(x) = \frac{\log(\frac{N}{N_x})}{\log(N)} \text{ , } N_x = |F(d,x)| \quad (2)$$

where S(*q*) is the set of all possible relaxations of *Q*, *F*(*d*,*x*) is the set of all files that match a relaxation *P* of *q* or x’s extension in *d*, and *N* is the total number of files in *d*.

IV. QUERY EVALUATION

We have adapted the Threshold Algorithm (TA) [12] in this approach. *Threshold Algorithm* uses a threshold condition to avoid evaluating all possible matches to a query, instead of focussing on identifying the *N* best answers. To overcome this problem a new algorithm was being proposed DAG_Path algorithm to handle our proposed relaxations.

Procedure for DAG-path(sNode)

```
1. Get the score of sNode into s and consider it as curNode.
2. Repeat
     a. get Depth of curNode into targetDepth.
     b. get first child of curNode as cNode.
     c. exit loop when score of cNode != s or cNode has no more childs.
     d. assign cNode to curNode.
3. For each n find the target depth and score.
     a. Perform bottom-up from n and identify ancestor node set X.
     b. For each m in X, set score to m.
     c. For each m in X.
          i. For each p on path n to m.
                  set score and set it as skippable.
          ii. if m is not skippable set it as skippable.
4. Apply RandomDAG on DAG_path to obtain desire path.
```

The algorithms DAG_Jump and randomDAG are referred from [14] which are used to obtain the accurate search results.

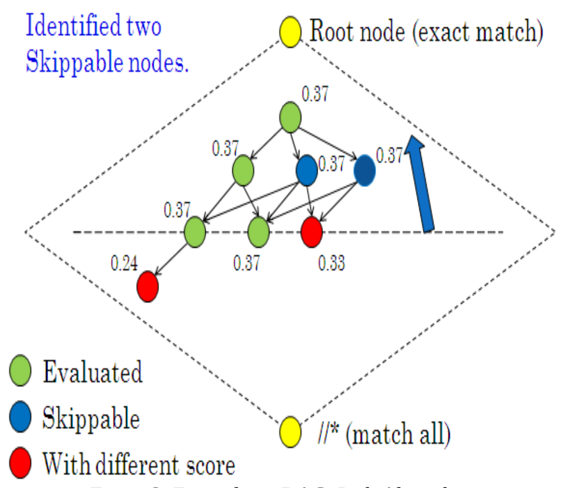

Figure 3: Example on DAG_Path Algorithm

We represent all possible matches of a query condition, along with the corresponding *IDF* scores for files that match, using a DAG structure. The DAG structure is created by applying query relaxations to the original query condition, the root of the DAG indicates the original exact query. Children nodes of a DAG node are more versions of the given query condition. The set of files matching a node should be equal or subsume of set of files matching its child nodes. The *IDF* score of a DAG node cannot be greater than the score associated with its parent node [1, 12]. As we expand the DAG and traverse it further away from the root, until the exact match of a node occurs. The most relaxed version of the query condition: //* matches all files and it has a score of 0.

There are several challenges in order to perform querying processing very efficiently:

• The set of all possible relaxations is query dependent and the size of the DAG grows exponentially with the query size, i.e., the number of path nodes in the query. As there is a chance to get the matched node at the beginning of the node. So there is no need to visit the remaining DAG. However, if the nodes are more than the index building and traversal techniques are critical issues.

• Query processing algorithms should be adapted to handle the proposed relaxations in an efficient manner.

V. EXPERIMENTAL RESULTS

Our study mainly focuses on the performance of the techniques which are used for searching a file. For each of the target file, we constructed queries based on the structure and content as shown in table 1.

Consider the data set which consists of 2345 photos which are organized in 115 directories. The target file is /photos/aa/batch14/Vishnu.png and structure condition is /photos/aa/batch14 which is considered as /p/a/b. Query1 consider only the content of file. Query 2 consists all the dimensions data. Query 3 consists of the filename with wrong file type. Query 4 consists of correct file type but with no date and path. Similarly other queries are considered based on different combinations of the structure and metadata.

Even the given condition is approximate the results are obtained for a given query which are not exact. By considering these techniques we can perform the search very effectively compare to the existing tools like google desktop search, spotlight. These results provide us a flexible search approach which is used to improve the accuracy rate of the result.

Query Evaluation Results						
Query Conditions						
Query	Content	Type	Date	Structure	Rank	Comments on queries
Target file:/photos/aa/batch14/vishnu.png						
Structure Condition: /photos/aa/batch14 consider as /p/a/b, 's' is incorrect						
Q1	Vishnu	-	-	-	16	Base Query
Q2	Vishnu	.png	11 -Nov- 14	/p/a/b	1	Correct values for all dimensions
Q3	Vishnu	.jpg	-	-	4	Incorrect File Type
Q4	Vishnu	.png	-	-	7	Correct File Type
Q5	Vishnu	-	Nov-14	-	6	Range for nov month
Q6	Vishnu	-	15-Nov-14	-	1	Incorrect Date
Q7	Vishnu	-	11-Oct-14	-	2	Incorrect Month
Q8	Vishnu	-	-	/b/a	1	Incorrect path
Q9	Vishnu	-	-	/p/a/b	1	Correct path
Q10	Vishnu	-	-	/p/a/s	1	Incorrect path
Q11	Vishnu	.png	11-Oct-14	/s/b	1	Incorrect path

Table 1: Target files-rank returned by a set of queries

VI. CONCLUSIONS AND FUTURE WORK

A unified framework is presented where the scores of all dimensions are unified into single framework. This framework provides flexible query processing on content and structure dimensions of personal information systems. We proposed query matching algorithms to efficiently evaluate ranked search queries over our unified framework.

This ranking framework is used to obtain the accurate results of a particular query over existing content based methods by leveraging information from both structure and content as well as relationships between the terms. Our work provides importance of search on approximation in personal information queries and opens important research directions for efficient and high quality search tools. In this paper, we have focused on files as the result unit. In the future, we will relax this restriction to allow for logical units of data to be returned. For example, a set of photos taken at the same time and location thread could constitute and be returned as a single logical entity.

In the future, we can extend our work by considering another dimension i.e, file context information. By using these techniques we make a fuzzy search approach practical for daily usage.

REFERENCES

[1] http://en.wikipedia.org/wiki/Personal_information_management.

[2] Google desktop. http://desktop.google.com.

[3] Apple MAC OS X spotlight. http://www.apple.com/macosx/features/spotlight.

[4] Y. Cai, X. L. Dong, A. Halevy, J. M. Liu, and J. Madhavan. Personal Information Management with SEMEX. In *Proc. of the ACM Intl. Conference on Management of Data (SIGMOD)*, 2005.

[5] J.-P. Dittrich and M. A. Vaz Salles. iDM: A Unified and Versatile Data Model for Personal Dataspace Management. In *Proc. of the Intl. Conference on Very Large Databases (VLDB)*, 2006.

[6] M. Franklin, A. Halevy, and D. Maier. From Databases to Dataspaces: a New Abstraction for Information Management. *SIGMOD Record*, 34(4), 2005.

[7] C. Peery, W. Wang, A. Marian, and T. D. Nguyen. Fuzzy Multi-dimensional Search in the Wayfinder File System. In Proc. of the International Conference on Data Engineering (ICDE), 2008.

[8] C. Peery, W. Wang, A. Marian, and T. D. Nguyen. Multi-Dimensional Search for Personal Information Management Systems. In Proc. of the International Conference on Extending Database Technology (EDBT), 2008.

[9] An XML Query Language. http://www.w3.org/TR/xquery/.

[10] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, and D. Toman. Structure and Content Scoring for XML. In Proc. of the International Conference on Very Large Databases (VLDB), 2005.

[11] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. FleXPath: Flexible Structure and Full-Text Querying for XML. In Proc. of the ACM International Conference on Management of Data (SIGMOD), 2004.

[12] R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. Journal of Computer and System Sciences, 2003.

[13] Ameilie marian, Wei Wang “flexible Querying of personal information” IEEE transactions and knowledge data ,2008.

[14] Wei Wang, Christopher Peery, Am´elie Marian, Thu D. Nguyen “Efficient Multi-dimensional Fuzzy Search for Personal Information Management Systems” IEEE transactions on knowledge and data engineering,2011.