

# CONFIDENTIALITY FOR MAINTAINING THE PERSONAL INFO BY USING MULTI FACETED FUZZY TECHNIQUE

**K. Aruna Kumari <sup>1</sup>, B. Venkat Suresh Reddy <sup>2</sup>**

<sup>1</sup>*M. Tech Scholar (CSE), Nalanda Institute Of Engineering & Tech. (NIET),  
Siddharth Nagar, Guntur, (India)*

<sup>2</sup>*Assistant Professor, Nalanda Institute Of Engineering & Tech. (NIET),  
Siddharth Nagar, Guntur, (India)*

## ABSTRACT

*With the detonation of the semi structured data users information stored or access in the personal information management system, there is a necessity of emerged search tool to often retrieve the heterogeneous information in simple and effective way. All the existing tools are supporting IR-style ranking on the textual part of the query. For filtering these techniques only consider the file structure (file directory) and file metadata (type of file). We proposing a new multi - dimensional search method, in this technique user perform fuzzy keyword searches, metadata and structure searches in addition to keyword conditions. This technique separately score each dimension after that it assembles all three individual dimension scores in to a meaningful incorporated score. We also prepare algorithms and indexes to effectively discovering the most related files those matched to multi- dimensional queries. We do a systematic experimental valuation of our method and represent that in non-content dimension the scoring context will expressively improve the exactness of ranking. We also represent that query processing policy do and scale well, makes our fuzzy search technique in every day usage.*

**Keywords: - Multi Dimensional Search, Personal Information Management System, IR-Ranking, Query Processing.**

## I. INTRODUCTION

In day to day storing the data in personal information management system is increasing. Due to dropping price of storage and growth of persistent capacity. This evolutionary storage is leads a problem of powerful searching tools to store and retrieve the heterogeneous information very simply and efficiently. Those tools will give both effective query processing capabilities and high quality scoring framework techniques. To perform keyword search many number of searching tools are introduced and find personal information in the local system, such as spotlight and Google desktop search. However, for the textual part of query these tools some form of ranking – equal to what has been completed in the information retrieval (IR) communal. A filtering conditions it only considering structure (file directory) and metadata of the file (e.g. file type, date). In the recent days research community shifted its focus to search on data spaces and personal information. It contains varied data information. However, in the case of commercial file system searching tools, which will works on IR- style key word query search, and uses other system files information to guide the keyword search. Keyword – search only is not at all sufficient, as demonstrated using the bellow examples. Example1: It considers user stores personal information in personal computer system. In calculation to the actual file content, the structure location

information (e.g. directory information), and file large amount of metadata information (date, file type) are also stored in the file system.

In the situation user might ask the query like:

```
[file type = *. doc AND  
    createDate = 21/10/2014 AND  
    content = "Proposal Draft" AND  
    structure = /docs/wayf/Propose].
```

The existing tools will give answer by returning the file type is \*. Doc and created on 21/10/2014 and have content matching to "Proposal Draft" (ranking) which are available in the directory /docs/wayf/Propose (filtering conditions), how close content matching to Proposal Draft by using some native text scoring mechanism. All the information except content using filtering conditions, files that are related to the query, but this do not satisfied the exact conditions was ignored. For the example a file type is \*. text created on 21/10/2014 available in the directory /docs/wayf/Propose comprises the content "Proposal Draft" will not returned. In many search situations we are debating that allowing the elasticity structure and metadata increasing the usefulness and quality of search results. For illustration, in example if the user may not remembering the exact file created date, but remember it was created around 25/10/2014. Like that initially the user want the file type is \*.doc, he may also want consider the related files which are related to the type of \*.text or \*.txt. At finally, the user might not remember where the file was exactly stored, at this movement by using the file size and structure conditions are not as filtering conditions, ranking conditions as part of query will returns best answers as part of the search results. In this paper we are going to propose, a novel technique, that lets the users to effectively do fuzzy keyword search through 3 different dimensions: structure, metadata and content. We will define independent IDF-based scoring approach for every dimension and represent unified scoring framework for multi-dimensional query system over the personal information system. To make finding scoring fuzzy matching more efficient we are presenting index construction optimization and new data structures. Since our proposed work is enhancement to different data space applications and queries, in this paper we will only concentrate on file search scenario, we will deliberate the granularity of the entire search results to single file in our personal system. May our technique was enhancement to the flexible query models, where the peace of data within the files can be returned as files.

Our detailed involvement includes

- We introduce IDF-based mechanism for scoring for content, structure and metadata, and a framework to combine the independent dimensions scores to united multi dimension scores.
- We inherit the current top-k query processing algorithms and recommend optimizations to increase access to structure dimension index.
- Experimentally we estimate our scoring framework and represent that our method has the expressively will improve the searching correctness with the existing filtering techniques.
- We solidly demonstrate the effectiveness of our technique optimizations on time of query processing and express that our approach will improve the query effectiveness and result in good scalability.

In both IR community and databases the discussion on integrating technologies from the two fields with structure based query results to combinations of content – only searches. Our mechanism provides a phase in this direction as they assembled DB-style structure with IR-style information score estimate scores.

## II. UNIFIED MULTI-DIMENSIONAL SCORING

In this segment, we are discussing about our unified framework for giving scores to the files depends how closely within dissimilar query dimensions they equal the query conditions. We discriminate three different scoring dimensions: The content for the conditions on the files textual related content, metadata for the conditions on the file which are related to system information, and structure for the condition for access the files which are based directory path. We represent the files structure information and related metadata in a XML documents. Here in the key based content conditions we are using easy version of XQuery to show structure and metadata conditions. Any file that is related one or more of the query conditions then it will get a prospective answer for the particular query. It is assigned by a score for every dimension depends on how much it matches to related query condition. Scores through the multiple dimensions are integrated in to a single total score for ranking of answers. Our scoring approach is based on clarification of scores IDF-based, as defined bellow. For each query condition we scores files depends on the last relaxed form of the condition that every file matches. Scoring throughout all dimensions is homogeneously IDF-based which allows us to meaningfully combine the multiple single dimensions to homogeneous multi dimension scores.

### 2.1 Scoring Content

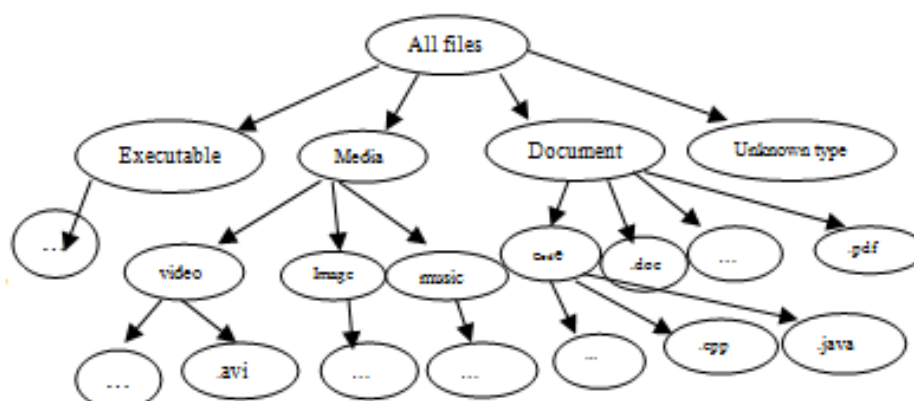
For content query conditions we use standard IR relaxation and scoring techniques. Especially we inherit the TF- IDF scoring formulas from a position of the art keyword search tool. These formulas as follows:

$$Score_c(Q,f) = \sum_{t \in Q} \frac{score_{ctf}(t,f) \times score_{cidf}(t)}{NormLength(f)} \quad (1)$$

$$Score_{ctf}(t,f) = \sqrt{No. \text{ times } t \text{ occurs in } f} \quad (2)$$

$$Score_{cidf}(t) = 1 + \log\left(\frac{N}{1 + N_t}\right) \quad (3)$$

Here Q is the query condition for content, f is the scored being file, and N denotes the total number of files,  $N_t$  specifies the number of files containing the word t, and NormLength (f) is a normalization factor that is a f's length function. Reminder that relaxation is an essential part of the above formulas from the score files that comprise a subset of terms in the query conditions.



**Fig. 1. Fragment of Relaxation DAG for File Type  
 (Extension) Media**

## 2.2 Scoring Metadata

To support scoring we are introducing classified relaxation technique for every type of searchable metadata. Relaxation levels for the file types represented in Fig 1, shown as a DAG. In this each leaf denoted as specific file type. Every internal node described as most general file types that is the combination of files like media of its children and hence its relaxation for its children. Containment is a key distinctive of this classified demonstration, which is if the group of files matching nodes that should be equals to the group of files matching to its child nodes. This confirm that the file score is matching to more relaxed query form condition is always less than or equal to the file score is matching a less relaxed form. We say that the condition of metadata is matches to a DAS node if the metadata node range is equals to or include the query condition. If the specified query condition contain a file “\*.cpp” which should match the nodes describing the files type “Code” and files of type “Document” so on. The query condition based on the time created a file, it will consider different time granularities like data, month and year, a node in the path the deepest matching node to the root of the DAG, then it shows the all the relaxation that how much we can score for the query condition. Similarly, if every file matches all the nodes in the DAG, then its equals to the files metadata values.

At finally, gives a query condition Q having the metadata condition M, with respect to Q the metadata score file f is computed as:

$$scoreMeta(Q, f) = \frac{\log\left(\frac{N}{nFiles(commonAnc(nM, nf))}\right)}{\log(N)} \quad (4)$$

Here N represents total number of files,  $n_m$  specifies the deepest node that matches M, Here nf is deepest DAG node matching with file f,  $commonAnc(x, y)$  is returns the nearest ancestor of the node x and y in the relaxation hierarchy, and  $nFiles(x)$  denotes returns number of files that matches node x.

The score is generalized by  $\log(N)$ , so with this the highest possible score value will give 1.

## 2.3 Scoring Structure

Most of the users organize their data using hierarchical structure. While searching for that file he needs to remember the components of the directory path or rough ordering rather than exact file location. Thus, considering some estimation on structure query conditions is required, because it allows the user memory power to search for particular even though he has some idea. Our structure scoring approach is enhancement to our existing work to XML structural query relaxations. Especially the node Reversal relaxation is introduced below to handle possible miss order pathname when specifying structure query condition in the personal file system. Let's assume the structure query conditions are given as file queries.

The relaxations are:

- **Edge Generalization:** this is used to relax a parent-child relationship to ancestor sliding relationship.
- **Path Extension:** which is used to extend the path P, all the files which are available in the directory contains p is answer for the query. Example extension path P to /a/b will be result as /a/b// \*.
- **Node Inversion:** it is used to change a node to within path query P. To denote possible changes, we are introducing a concept group of nodes as path, where the node place is fixed and nodes may be change. The variation is applicable to its beside nodes or node group except for \*.nodes and root. The permutation associations' adjacent nodes or group of nodes as a single node group while providing the relevant order on edges in p.

- **Node Deletion:** it is used to completely drop a node from the path, which is applicable to query path or node group, but it is not applicable for delete the root node and \*.nodes.

To delete a node  $n$  from the path query  $P$ .

- $n$  should be a leaf node then only  $n$  will be dropped from  $P$  and  $n$  must extended with  $//*$ . This is to guarantee repression to correct answer to  $P$  in the group of answers to  $P'$

- if  $n$  is an internal node in the  $p$  at that time the parent node ( $n$ ) and child node  $n$  should be connected in the  $P$  with  $//$ .

## 2.4 Score Aggregation

We combine the above mentioned single dimension score in to uniformed multi dimension score to give a costumed ranking values relevant to the multi-dimensional query. To accomplish this we prepare a query vector  $V_Q$  structure with the exact value is 1 for every dimension and a file vector,  $V_F$ , with respect to  $Q$  containing of the single dimension scores for a file  $F$ . We then perform  $V_F$  projection on  $V_Q$  and the length of the vector result is used as aggregated score file  $F$ . With its existing form, this is simply a lined combination of component scores using equal weighing.

## III. QUERY PROCESSING

We are inheriting an existing algorithm is known as threshold algorithm (TA) to do query processing. To avoid all the possible matches to a query TA uses a threshold condition, concentrating rather to identifying  $k$  best answers. It takes input several sorted lists, each sorted list contains the system object according to their particular attributes its sorted in ascending order, and dynamically access the sorted lists until the thresh hold reaches to find the  $k$  best answers. To retrieve independent attribute scores TA trusts on random access. Consider sorted accesses that are sorted list declared in the above, needs the files to be back in the descending order based on the scores for the particular dimension. Random access needs any files particular computation score of any given file. Random access appears when Thresh hold algorithm find a file from a specified list corresponding to some dimension, then it require the score of files which are available in all the dimensions to perform the computation its unified scores. To use TA with our setup, indexed structures and algorithms needs to support both random access and sorted each of our three dimensions.

### 3.1 Evaluating Content Score

We use the existing TF-IDF mechanisms to score the content dimension this is performing according to mention above. Random accesses are supported through the standard reversed list implementations. Here, for every query term, we can easily see the term frequency in entire file system as well as in a single file. We are doing sorted access by keeping the inverted lists in sorted order that is group of files having the particular term, keep them in to a sorted order according their TF scores, and do normalize the file size.

### 3.2 Evaluating Metadata Scores

By using an appropriate relaxation DAG index we implement sorted access for a metadata condition. First matching content is identified by the deepest DAG node that matches in the given condition. Exact matches are fetched from  $N$ 's leaf side by nodes, the approximate matching nodes are crossing up DAG to consider more estimated matching things. Each and every parent contains larger values than its children, it specifies the matches are done in decreasing order of metadata scores. Equal to content dimension, we are using TA algorithm to fetch the values recursively in sorted for respected queries that having multiple metadata conditions.

### 3.4 Evaluating Structure Score.

The structure score for a particular file depends on matches to a query condition in which directory exactly the file was stored. All the files within in the same directory have same structure. To calculate the structure score of a file  $f$  in the directory  $d$  and the structure condition  $P$  for a given query, we first need to identify all the directory paths with including of  $d$  that match with  $P$ .



## IV. CONCLUSION

We are proposing a scoring technique to multi-dimensional queries on personal information management systems. Especially, we are defining structure and metadata information and we proposed IDF-based scoring technique for metadata, content and structure query condition. This constant Scoring allows the individual dimension scoring in to combine. We also need to design indexing structure and query processing optimizations and dynamic index construction to effective evaluation of multi-dimension queries. We executed and calculated scoring frameworks and query processing techniques. Our evaluation represents that our multi-dimensional score aggregation mechanism reserve characteristics of individual scores and contains significantly improve the accuracy of ranking. We also represents that our indexes and optimizations are made multi-dimensional search is efficient and in daily search use.

## REFERENCES

- [1] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *Proc. Of the First Biennial Conference on Innovative Data Systems Research (CIDR'03)*, 2003.
- [2] S. Amer-Yahia, P. Case, T. R'olleke, J. Shanmugasundaram, and G. Weikum. Report on the DB/IR panel at SIGMOD 2005. *SIGMODRecord*, 34(4), 2005.
- [3] S. Amer-Yahia, S. Cho, and D. Srivastava. Tree Pattern Relaxation. In *Proc. Of the Intl. Conference on Extending Database Technology(EDBT)*, 2002.
- [4] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, and D. Toman. Structure and Content Scoring for XML. In *Proc. Of the Intl. Conference on Very Large Databases (VLDB)*, 2005.
- [5] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit. FleXPath:Flexible Structure and Full-Text Querying for XML. In *Proc. Of the ACM Intl. Conference on Management of Data (SIGMOD)*, 2004.

## AUTHOR PROFILE

	<b>K. Aruna kumari</b> is currently pursuing M.Tech in the Department of Computer Science & Engineering, from Nalanda Institute of Engineering & Technology (NIET), siddharth Nagar, Kantepudi(V), Sattenapalli (M), Guntur (D), Andhra Pradesh , Affiliated to JNTU-KAKINADA.
	<b>B. Venkat Suresh Reddy</b> working as Assistant Professor at Nalanda Institute of Engineering & Technology (NIET), siddharth Nagar, Kantepudi(V), Sattenapalli (M), Guntur (D), Andhra Pradesh , Affiliated to JNTU-KAKINADA.