

Fuzzy Multi-Dimensional Search in the Wayfinder File System

Christopher Peery, Wei Wang, Amélie Marian, Thu D. Nguyen
Department of Computer Science, Rutgers University
110 Frelinghuysen Rd, Piscataway, NJ 08854, USA
{peery, ww, amelie, tdnguyen}@cs.rutgers.edu

Abstract— With the explosion in the amount of semi-structured data users access and store, there is a need for complex search tools to retrieve often very heterogeneous data in a simple and efficient way. Existing tools usually index text content, allowing for some IR-style ranking on the textual part of the query, but only consider structure (e.g., file directory) and metadata (e.g., date, file type) as filtering conditions. We propose a novel multi-dimensional querying approach to semi-structured data searches in personal information systems by allowing users to provide fuzzy structure and metadata conditions in addition to traditional keyword conditions. The provided query interface is more comprehensive than content-only searches as it considers three query dimensions (content, structure, metadata) in the search. We have implemented our proposed approach in the Wayfinder file system. In this demo, we will use this implementation to both present an overview of the unified scoring framework underlying the fuzzy multi-dimensional querying approach and demonstrate its potential in improving search results.

In such a scenario, it is possible to ask the query:

```
[ filetype=*.doc AND  
  createDate=03/21/2007 AND  
  content="proposal draft" ]
```

*Current tools would answer this query by returning all files of type *.doc created on 03/21/2007 (filtering conditions) that have content similar to “proposal draft” (ranking expression), ranked based on how close the content matches the text “proposal draft” using some underlying text scoring mechanism. Because the date and file type are used only as a filtering conditions, files that are very relevant to the content search part of the query but which do not satisfy the filtering conditions would not be considered as valid answers. For example, *.tex documents created on 03/19/2007 would not be returned.*

We believe that allowing flexible conditions on structure and metadata can significantly increase the usefulness of search engines in many file search scenarios. For instance, for the Example 1 query, the user might not remember the exact creation date of the file of interest but remembers that it was created *around* 03/21/2007. The use of filtering conditions would require specifying a range of dates that would (hopefully) include the target file; an incorrect range would preclude the target from being returned as a relevant result. On the other hand, when the date is used as a ranking condition, it becomes unnecessary to specify such a range as all files would be considered with those having dates close to the query condition receiving high relevance scores and those further away receiving progressively lower scores.

The challenge is then to adequately score the files present in the file system by taking into account flexibility in the textual component *together* with some flexibility in the structural and metadata components of the query. Once an adequate scoring mechanism is chosen, efficient algorithms to identify the best files that match the query, *without considering all the files in the file system*, are also needed.

In our work, we propose a novel approach that allows users to provide fuzzy conditions on three query dimensions: content, metadata, and structure. We have designed an *IDF*-based unified scoring framework that allows the computation of a single score across the above three query dimensions for each relevant file. We have implemented a prototype of our unified scoring framework together with a top-*k* retrieval algorithm based on [6] in the Wayfinder file system [7].

I. INTRODUCTION

With the explosion in the amount of data users access and store there is a need for complex search tools to access often very heterogeneous data in a simple and efficient way. Numerous third-party search tools have been developed to perform keyword searches and locate personal information stored in file systems, e.g., [1], [2]. These tools usually index text content, allowing for some *ranking* on the textual part of the query, similar to what has been done in the Information Retrieval (IR) community, but only consider structure (e.g., file directory) and metadata (e.g., date, file type) as *filtering* conditions.

Recently, the research community has turned its focus on search over Personal Information and Dataspaces [3], [4], [5], which consist of heterogeneous data collections. However, as is the case with commercial tools, these works focus on IR-style keyword queries and use other system information only to guide the keyword-based search. Keyword-only searches are often insufficient, as illustrated by the following example:

Example 1. *Consider a user saving personal information in the form of files in a file system. In addition to the actual file content, structural location information (e.g., directory structure) and a large amount of metadata information (e.g., access time, file type) are stored alongside the file content in the file system.*

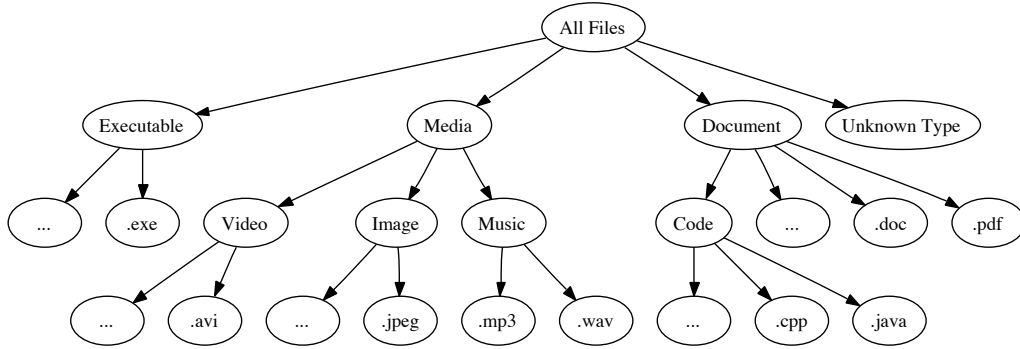


Fig. 1. The File Type Metadata DAG.

In this paper, we will first present an overview of our scoring framework, briefly explaining the *IDF*-based scoring function for each of the three dimensions, the indexing data structures required to efficiently compute the scores, and our approach to aggregating the three single-dimensional scores into one unified score. The interested reader can find more details about our approach in [8]. We then briefly describe the Wayfinder file system to provide the context for our implementation. Finally, we conclude with an overview of the demonstration given at the conference.

II. SCORING

In this section, we give an overview of our unified scoring framework and the supporting indexing structures. As already discussed, we seek to support fuzzy search along three dimensions: *content* for conditions on the textual content of the files, *metadata* for conditions on the system information related to the files, and *structure* for conditions on the directory path to access the file.

We score relevant files for each dimension using an *IDF*-based approach. Traditionally, the *IDF* score of a document for a keyword in the IR world is a function of how many documents contain the keyword [9], generally of the form $\log(1 + \frac{N}{f_t})$, where N is the total number of documents in the system and f_t is the number of documents containing the respective term. The content *IDF* scoring strategy has been widely adopted in IR systems as it considers the data distribution to assign scores. We extend this idea to each of our search dimension and assign a score to a file in a query dimension based on how many files match the query condition for that dimension. We can then meaningfully aggregate multiple single-dimensional scores into a unified multi-dimensional score as described in Section II-D since they are uniformly *IDF*-based.

A. Scoring Content Conditions

We employ standard indexing and scoring techniques from the IR literature [9] for query conditions involving textual content. Specifically, we have implemented inverted indexes for identifying files containing query terms and use a $TF \cdot IDF$ scoring strategy for scoring the relevance of these files to the text conditions.

B. Scoring Metadata Conditions

All file systems maintain a range of metadata information alongside file content. Such metadata may include file sizes, file owners, and various file timestamps (e.g., date created and date last modified). File extensions can also hint at the corresponding file types. Users often want to enhance their query with metadata conditions (e.g., file was accessed last week, file is a pdf document), but may not accurately remember the exact metadata values for which they are looking. Therefore, allowing for some approximation in metadata conditions is desirable.

We introduce the notion of *metadata relaxations* to retrieve and score approximate matches to metadata query conditions. We use a DAG indexing structure for each type of searchable metadata to support both the retrieval and the scoring. For example, Figure 1 represents (a subgraph of) the indexing DAG for file types. Each leaf of this DAG represents a specific file type (e.g., pdf files) and contains a count as well as references to all files of that type. Each internal node represents a more general file type that is the union of the types of its children (e.g., *Media* is the union of *Video*, *Image*, and *Music* types) and thus is a relaxation of its descendants. Correspondingly, each internal node contains the sum of the file counts of its descendant leaves. Note that the count maintained at each internal node is thus guaranteed to be greater than or equal to the count at any of its children.

Given a specific metadata condition, the path from the matching leaf to the root of the DAG indexing structure of that metadata type then represents all of the approximations that we can score for that condition. For example, given the DAG shown in Figure 1, the query condition `filetype=*.doc` would have an exact match, the leaf corresponding to type `.doc`, and two approximate (relaxed) matches, the internal nodes corresponding to the types *Documents* and *All Files*. Our *IDF*-based scoring approach then scores a file that matches one of these three approximations of the query condition inversely proportional to the number of files that matches that approximation. Continuing the example, for the condition `filetype=*.doc`, files of type `.doc` would have the highest score as they are exact matches to the condition. Files of type *Document* other than type `.doc` (`.txt`, `.pdf`, and *Code*) would be

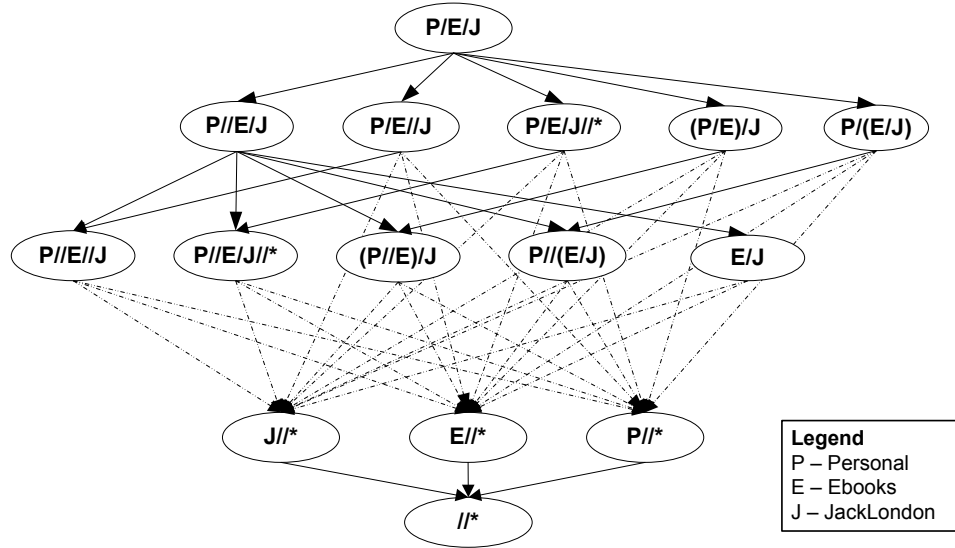


Fig. 2. The structure DAG for the structural query condition *Personal/Ebooks/JackLondon*. Solid lines represent parent-child relationships. Dotted lines represent ancestor-descendant relationships, with intermediate nodes removed for simplicity of presentation.

assigned a lower score. Finally, files of type *All Files* other than type *Document* (all remaining files in the system) would be assigned yet a lower score.

For queries involving multiple metadata conditions, e.g., the query in Example 1 contains a condition on date and a condition on filetype, the individual condition scores have to be aggregated to produce a unified metadata score. Our aggregation approach is exactly the same as the aggregation approach for deriving the final unified multi-dimensional score, which is discussed in Section II-D.

C. Scoring Structure Conditions

Most users use a hierarchical directory structure to organize their files. When searching for a particular file, a user may often remember some of the components of the containing directory path and their approximate ordering rather than the exact path itself. Thus, allowing for some approximation on structure query conditions is desirable because it allows users to leverage their partial memory to help the search engine locate the desired file.

Our structure scoring strategy extends prior work on XML structural query relaxations [10], [11]. In particular, we use several types of structural relaxations, some of which were not considered in [10], [11], to handle the specific needs of user searches in a file system. These relaxations are:

- **Edge Generalization** is used to relax a parent-child edge to an ancestor-descendant edge. This transforms a path query from $/a/b$ to $/a//b$.
- **Path Extension** is used to extend a path P such that that all files within the directory subtree rooted at P can be considered as answers. This transforms a path query from $/a/b$ to $/a/b//*$.
- **Node Deletion** is used to drop a node from the path query. This transforms a path query from $/a/b/c$ to $/a//c$.

- **Node Inversion** is used to permute nodes within a given query. This transforms a path query from $/a/b/c$ to $/a/(b/c)$ allowing for both the original query as well as $/a/c/b$.

As proposed in [10], we use a DAG to represent all possible structural relaxation of a path query condition. Figure 2 shows an example relaxation DAG for the structure query condition *Personal/Ebooks/JackLondon*. This DAG is rooted at the exact query condition itself, with each non-root node representing a relaxed form of the exact query condition. Note that this *structure* DAG is a representation of the query and all possible relaxed forms of that query given the above relaxation operations, rather than an index of data in the file system like the indexing metadata DAG in Figure 1.

Given a structural query condition, we dynamically compute the structure DAG for that query condition. Then, to compute the *IDF*-based scores of files relevant to that query condition, we must compute the number of files that match each node of the DAG.¹ In this structure DAG, the number of files matching a node must be greater than or equal to the number of files matching the node's parent, since the child is a more relaxed form of the query condition than the parent. Thus, scores decrease for files that match nodes further down in the DAG.

We have designed several algorithms and indexing structures for efficiently and lazily generating the DAG and computing scores of matching files. Discussion of these algorithms and indexing structures are beyond the scope of this paper. We again refer the interested reader to [8].

¹When we say that a file matches a structure query condition (or a relaxed form of the condition), what we really mean is the file is contained in a directory that matches the condition.

D. Score Aggregation

We now need to aggregate the above single-dimensional scores into a unified multi-dimensional score to provide a unified ranking of files relevant to a multi-dimensional query. To do this, we construct a query \vec{V}_Q having a value of 1 (exact match) for each dimension and a file vector, \vec{V}_F , consisting of the scores of the individual query dimensions. We then compute the projection of \vec{V}_F onto \vec{V}_Q and the length of the resulting vector is used as the aggregated score.

III. IMPLEMENTATION

As already mentioned, we have implemented a prototype of the above multi-dimensional scoring framework together with a top- k retrieval algorithm in the Wayfinder file system. We chose Wayfinder as the hosting platform for our implementation mostly based on convenience. Wayfinder is a user-level file system, which makes it significantly easier to modify. In addition, Wayfinder already implements a $TF \cdot IDF$ -based content search engine (along with the necessary supporting indexes).

In this work, we have extended Wayfinder to include the following data tables and indexes to support the processing of query conditions in the metadata and structural dimensions:

Metadata:

- A metadata table, where each row contains the typical metadata, e.g., owner, size, date of creation, etc., for each file in the system. (In current file systems, this information is typically dispersed throughout the file system, making it difficult to find files that match specific metadata query conditions.)
- Five indexes of the metadata table to support efficient searches on the size, type, date-of-creation, last-modified-time, and last-accessed time attributes. Each of these index corresponds to a DAG similar to the one presented for file type in Section II-B.

Structural:

- Two name-binding tables to hold “parent directory \rightarrow children” and “child \rightarrow parent directory” bindings.
- An inverted index that maps terms to full directory pathnames that contain these terms for quickly finding directories that match a structural query conditions and its relaxed forms.

Wayfinder is written in Java. Tables and indexes are persistently stored using the Berkeley DB [12]. Search queries are specified using a small subset of the XQuery language (although in the demo, we will be using a GUI that hides this query language interface).

IV. DEMONSTRATION

We will demonstrate our system by executing a number of queries over a data set of approximately 26,000 files that comprise a subset of a real user’s personal file system. This data set contains a subset of the user’s actual hierarchical namespace together with file of various file types, including

media files, image files, various document files, source code, and email messages. The demonstration is designed to explore the benefits of supporting fuzzy conditions in the metadata and structure dimensions in addition to the traditional content dimension. Query construction and result presentation will be done using a web-based interface that will access a running instance of the Wayfinder file system.

Motivation and Problem Statement The first portion of the demonstration will involve a brief overview of the motivation behind our work, the design of our scoring infrastructure, and its integration in the Wayfinder file system. Several example queries will be presented to illustrate the limitations of current search methods that we are attempting to address in our work.

Query Example The second portion of the demonstration will show the benefits of our fuzzy multi-dimensional approach over content-only queries. In particular, we will contrast search results from queries employing non-content data as filtering conditions with results returned by our system. Our experimentation has shown that relevance ranking based on fuzzy matching against multi-dimensional queries can significantly improve the rank of a target file over content-only relevance ranking. For example, in our demonstration, we will show how the use of fuzzy matching in the structure and/or metadata dimensions in the search for a particular file can improve the ranking of that file from 49 to one of the top 10. For completeness, we will also present several queries in which the use of non-content query conditions actually degrades our search results and discuss the reasons why.

REFERENCES

- [1] “Google desktop,” <http://desktop.google.com>.
- [2] “Apple MAC OS X spotlight,” <http://www.apple.com/macosx/features/spotlight>.
- [3] Y. Cai, X. L. Dong, A. Halevy, J. M. Liu, and J. Madhavan, “Personal Information Management with SEMEX,” in *Proc. of the ACM International Conference on Management of Data (SIGMOD)*, 2005.
- [4] M. Franklin, A. Halevy, and D. Maier, “From Databases to Dataspaces: a New Abstraction for Information Management,” *SIGMOD Record*, vol. 34, no. 4, 2005.
- [5] J. Dittrich and M. A. V. Salles, “iDM: A Unified and Versatile Data Model for Personal Dataspace management.” in *Proc. of the International Conference on Very Large Databases (VLDB)*, 2006.
- [6] R. Fagin, A. Lotem, and M. Naor, “Optimal Aggregation Algorithms for Middleware,” *Journal of Computer and System Sciences*, 2003.
- [7] C. Peery, F. M. Cuenca-Acuna, R. P. Martin, and T. D. Nguyen, “Wayfinder: Navigating and Sharing Information in a Decentralized World.” in *Databases, Information Systems, and Peer-to-Peer Computing - Second International Workshop, (DBISP2P)*, 2004.
- [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] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, Inc, 1999.
- [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] Sleepycat Software, “Berkeley DB,” <http://www.sleepycat.com/>.