A Proposal for Exhaustive Search on Desktop Data

Mamta Kayest* and S. K. Jain
National Institute of Technology, Kurukshetra 136 119, India

Abstract
Data on one’s personal computer (PC) is continue to increase day by day and it has made need of managing personal desktop data as an active area of research. Managing desktop data includes an efficient way of searching and retrieving desired data and information from it. Several search engines and search tools are developed to provide search on the desktop data. In this paper, we present a solution for providing exhaustive search on one’s heterogeneous desktop data.

1. Introduction
Managing personal data is an emerging area of research because of the availability of large amount of digital data. Desktop data is a form of personal data available on one’s personal computer system. Desktop Search is a specialized instance of the Information Retrieval (IR) technology and provides search over personal data items on one’s desktop. Various Desktop Search Engines (DSEs) such as Google14, Yahoo!23, Coppenic8, X!22 manage one’s personal desktop data with an extension to email servers. A DSE uses crawler programs to extract data from various data sources that is indexed for further processing to get desired information. In this paper, we design a desktop search system for managing personal desktop data.

The proposed system makes search for files & folders and also retrieves partial contents of a semi-structured and unstructured data files. Here, a semi-structured and unstructured data file refer to an XML file and a text data file respectively. Files are searched on the basis of their properties. A user can search a file on the basis of one or more file’s properties such as file name, last access date, last-modified date, creation date, extension and size (KB). For folders one can search on the basis of properties including folder name, creation date, modification date, and access date. Partial content retrieval from semi structured data text files have also been included in the proposal as retrieving partial contents from files is an active research issue in managing desktop data19.

The next section of the article discusses problem definition and related work briefly, Section 3 and 4 presents a Data Flow Diagram (DFD) and algorithm for the proposed system respectively and Section 5 concludes the paper with future work.

*Corresponding author. Tel.: +91 -8221906216.
E-mail address: mamtakayest.nitkkr@gmail.com
Personal data refers to data related to one who accessed, owned and maintained it during his/her lifetime. Personal data consists of a variety of data including images, videos, semi-structured data, text data, emails, and so on. This data may be available in both centralized and distributed forms. Desktop data is an example of centralized personal data; various desktop search engines and tools have been developed for managing and improving search over data. Feldsper, Phlat, Semex, XSearcher, etc. are few examples of search system that retrieve information using associations among data. Google desktop search, Yahoo!, Corpernic Desktop Search, Windows search and many more desktop search engines are in use for searching data from desktop systems. These search systems have been compared on various parameters. The DSEs works on the principles of file systems of the underlying operating systems. One of the limitation with the DSEs is that they do not support the retrieval of partial contents from files. To search through a DSE approach, user first enters the search query to the search engine, and the search engine looks up the indexed database for getting desired results. Crawler programs are used by the DSEs for crawling and extracting information. This information is then further used by an indexer to create an indexed database. The problems with DSEs are that they lack in providing partial retrieval of information, supports simple keyword queries only; hence, do not support complex queries, no support for semantic integration, and takes initial indexing time once installed. Modelling and querying of heterogeneous desktop data further increases the complexity for developing a desktop search system. It explores a new research dimension for personal data management. In 2006, a graph data model has been proposed for modeling and providing uniform view over the personal data namely iMeMex Data Model (iDM) and to query over the uniform view a query language iMeMex Query Language (iQL) is used, which is complex to understand by a novice; here, it is expected from the users that they have knowledge of the underlying structure of the personal data. Recently in 2015, a framework is introduced for accessing heterogeneous data with different methods for query answering. Similarly, various other methods have been proposed to query over XML data. In our previous work, we proposed a desktop search system for searching desktop data for various file types and this paper is an extension of our previous work.

3. A Data Flow Diagram and Description of the Proposed System

This section describes the design of the proposed search system in detail. The desktop search system offers metadata based search on files & folders, and content based search on XML and text file. Figure 1 depicts a context diagram of the system which is further decomposed in Fig. 2 and Fig. 3. User submits a query to the desktop search system which in turn interacts with the file system of underlying operating system to get required information.

The system processes a query and return results to the user. The first level DFD of the context diagram is shown in Fig. 2. Fig. 3 exhibits second level DFD of the context diagram. Figure 2 contains four modules namely (1) execute file search, (2) execute folder search, (3) execute XML search, (4) execute text search. The names of each module also exhibit the functionalities to be implemented by them.

Figure 3 exhibits the further details of the data and modules as shown by the DFD in Fig. 2. Here, process input data module resolves a query based on its category and forwards to one of the four modules as shown in Fig. 3. Decomposition of input data for the four modules have also shown in Fig. 3. Individual functionality of each module in the DFD shown in Fig. 3 supports a modularization criteria which appears appropriate here. The system offers several search options to the users and the same can be summarized here as follows:
Fig. 2. Level 1 DFD.

Fig. 3. Detailed DFD of the Proposed System.
• Searching files on the basis of metadata including file name, size, creation date, last modified date, last access date, and file type.
• Searching folders on the basis of metadata including folder name, creation date, last modified date, and last access date.
• Retrieve partial contents from XML files using tags, sub tags, and contents with tags.
• Retrieve partial contents from text files and word documents using text pattern.

For a query on files/folders a user mention path for search. If user does not mention the path then the entire disk will be searched by the system. Once query criterion is entered, the file system is searched for the matches and the matched results are displayed to the user. For making search over XML file, the system can search file on the basis of tag name, contents of tags or both. The system is expected to process a variety of queries, few of the sample queries are given below:

1. Search for files where the file size is 500 MB from d drive.
3. Search drive g for folders which are modified on January 10, 2016.
4. Search folder named Nisha from f drive.
5. Search text files containing word dataspace and the file is modified on January 11, 2015.
6. Search contents of tag <employee> from XML files.

4. Algorithms for Searching Desktop Data

In this section, we discuss algorithms of the proposed desktop search system. These algorithms include algorithm for searching files on basis of their metadata, and algorithms for partial content retrieval from XML and text files. Algorithm 1 describes search over files and Algorithm 2 and 3 describes contents based search on XML and text files respectively.

Algorithm 1. Search for File using Metadata

| Step 1. Start |
| Step 2. Enter one or more search criteria |
| Step 3. Set path to start search |
| Step 4. Read files from the selected path |
| Step 5. Process files one by one |
| Step 6. Read properties of file |
| Step 7. If file size is the search criteria then |
| Ignore file not in range and continue |
| End if |
| Step 8. If file creation date is the search criteria then |
| Ignore file not in range and continue |
| End if |
| Step 9. If file last modification date is the search criteria then |
| Ignore file not in range and continue |
| End if |
| Step 10. If file last access date is the search criteria then |
| Ignore file not in range and continue |
| End if |
| Step 11. If file name is the search criteria then |
| Compare the name of file and continue |
| End if |
| Step 12. If file type is the search criteria then |
| Compare extension of file and continue |
| End if |
| Step 13. If entered query map to the file then |
| Display results |
| End if |
| Step 14. Check if selected file is a folder/directory then |
| Apply search on that folder recursively for search |
| Step 15. Stop |
Algorithm 2. Content Search from Text Files having Extensions .doc, .docx, .txt, .rtf, and .pdf

Step 1. Start
Step 2. Set the path to make search on
Step 3. Input pattern to search from file
Step 4. [Process search in word documents]
    If file name contains .doc or .docx then
        Create object of WordExtractor
        Read file contents line by line
        If pattern mapped from line then
            Count frequency of mapped pattern in variable count
            Move to next line
        End if
        If count=0 then
            Display results and set count to 0
        End if
    End if
Step 5. [Process search in .txt file]
    If file name contains .txt then
        Generate a BufferedReader for processing text file
        Read file line by line
        If pattern mapped from line then
            Count frequency of mapped pattern in variable count
            Move to next line
        End if
        If count=0 then
            Display results and set count to 0
        End if
    End if
Step 6. [Process search in .pdf file]
    If file name contains .pdf then
        Use pdf document object for pdf file processing
        Read the pdf text contents
        Check if query pattern exist in pdf text then
            Count the frequency of matched pattern in variable count
            Move to next line
        End if
        If count=0 then
            Display results and set count to 0
        End if
    End if
Step 7. [Process search for .rtf file]
    If file name contains .rtf then
        Generate object for rtf file processing
        Extract document text
        Check if the text contains text query pattern then
            Count number of occurrences of matched pattern in count
            Move to next line
        End if
        If count=0 then
            Display results
        End if
    End if
Step 8. Stop

Algorithm 3. Search on XML File

Step 1. Start
Step 2. Set path
Step 3. Enter tag name or contents or both to search from file
Step 4. Parse the XML file
Step 5. If tag name is the search criteria then
    Process all nodes to check the tag present as node and display contents of the matched tag(s)
Else if query contents is the search criteria then
    Process all nodes to check the tags that contains the query contents and display contents of the matched tag(s).
Else if both tag name and contents are the search criteria then
    Process all nodes to check both the tag name and contents and display results of matched pattern
End if
Step 6. Stop
5. Conclusions and Future Directions

In current scenario, management of user’s personal data is an emerging issue as personal data is both distributed and heterogeneous in nature. Various tools and desktop search engine such as Google, Yahoo!, etc. have been developed to provide search over the desktop data. Problem with these search engines is that they take extra indexing time prior starting their processing and also do not provide partial content retrieval from files. In this paper, we extended our previous work\textsuperscript{16} for more file types and proposing a solution for managing desktop data. The system offers search over desktop data using metadata and also supports partial content retrieval from files (XML files, text files). The implementation of the proposed system is in progress. Implementation and extension of our work for adding features such as use of wildcard characters for file name in searching are our immediate future work.

References