International Journal of Research in Engineering, Science and Management
Volume-1, Issue-11, November-2018
www.ijresm.com | ISSN (Online): 2581-5792

Desktop Exhaustive Search Algorithm for Offline Systems with N File Extensions

Abhishek Sharma[1], Swapnil Singhal[2]
[1]*Research Scholar, Department of Computer Science, Jaipur Institute of Technology, Jaipur, India*
[2]*Associate Professor, Department of Computer Science, Jaipur Institute of Technology, Jaipur, India*

***Abstract*: This paper is intended to present an algorithm of searching files in file space system of an offline system to make search easy and faster as compared to the existing file search space algorithms. DES systems are highly time consuming and are very complicated as they require to visit each and every file space of the computer system and hence uses a lot of memory resources of a system. In this paper the algorithm used for the DES operation is based on parallel computing and hence requires less time to search any workspace.With the rapid rise in computer hard drive capacity, the amount of statistics deposited on personal computers as digital photos, text files, and multimedia has amplified considerably. It has become time consuming to search for a specific file in the sea of files on hard disks. This has headed to the growth of numerous desktop search engines that help trace files on a desktop efficiently. In this paper, the presentation of five desktop search engines, Yahoo, Copernic, Archivarius, Google, and Windows are estimated. A recognized dataset, TREC 2004 Robust track, and a set of files demonstrating a classic desktop have been used to achieve comprehension experimentations. A typical set of evaluation procedures including recall precision averages, document level precision and recall, and precise exactness and recall over recovered set are used. The estimations performed by a typical evaluation package deliver an exhaustive presentation comparison of the desktop search engines by illustrative statistics retrieval procedures.**

***Keywords*: Parallel Computing, Data Storage, Data Mining, File Search Indexing, File Extension Indexing**

1. Introduction

Desktop search engines, are also called localized search engines which only search within the given the user desktop hard disk search area and are independent of the internet services, index and search files in a personal computer (pc). The data search algorithms are highly important and complex as the algorithm constructed are very critical and keeps themselves busy with continuous switching between the directories. To conduct file searches on the pc's hard drive, presentation of the desktop search engine in relations of information retrieval (ir) procedures, e.g. precision and recall, play a significant role in computing the correctness of the examine outcomes. Numerous corporations have unconfined their forms of desktop search engines like Microsoft Windows desktop search, Yahoo desktop search, Copernic desktop search, Google desktop search, Archivarius 3000, and Ask Jeeves. Of all these obtainable tools, the presentation of five, Windows desktop search, Google desktop search, Archivarius, Yahoo desktop search and Copernic desktop search are assessed and examined in this paper using standard Information Retrieval assessment procedures. The document Starts from here. And the section 2 continues accordingly.

2. Techniques used in past

The five desktop search engines are measured on the following standard and procedures on TREC documents:

- Recall-precision average
- Document-level precision
- R-Precision
- Document-level recall
- Mean Average Precision (MAP)
- Exact precision and recall over retrieved set
- Fallout-recall average
- Document-level relative precision
- R-based precision

These procedures are nominated for appraisal as they will be accountable for understandings into how desktop search engines incrementally improve documents and shape their outcome sets for a cluster of requests and the effect that has on the exactness of ultimate query outcome sets. The estimation on classic user desktop documents is complete with average recall and average precision procedures over all queries. We appraise the succeeding desktop search engines founded on the overhead principles. Microsoft's Windows desktop search (WDS) is closely interlinked with the Operating System versions from Microsoft be it for the desktops, tablets or for smartphones. Time taken by them to make a search is very huge as they stores the complete index of the system and scans every directories and the sub directories of the system hence took longer time, also it does not allow any search space customization. Yahoo desktop search (YDS) is established on X1 desktop search. YDS take a "reductive" methodology to demonstrating results. It benefits in selecting directory which only encompasses the content that has been certain like files, emails, IMs, contacts and to set distinct indexing options for each type of content. YDS delivers fine filtered and detailed control over indexing possibilities like specifying the folders, that must be indexed or the file kinds that can be indexed. YDS permits saving queries for future use, and shaping these hunts together with the general requests in the hunt pane. Copernic desktop search (CDS) agrees files types to be selectively indexed. Operator can select to index video, audio, images, and documents. It permits third party designers to create plug-ins that allow new file type indexing. For commercial use, Coveo, a spin-off company from Copernic, delivers enterprise desktop search produces with improved safety, manageability, and network capability. Google desktop search tool permits users to scan their own computers for statistics much the same way as they do for using Google to search the Web. Out of the much topography this tool delivers, noteworthy features include recurring search fallouts concise and characterized into dissimilar reinforced file types with a total count of competitions related with each type. Archivarius desktop search is a full-feature application designed to search documents and e-mails on the desktop computer as well as network and detachable disks. It permits files to be investigated on many progressive characteristics like alteration date, file size, and scrambling.

3. Literature survey

Search engines acts as medium between a user and a documents present in a system disk space. Without desktop search no information can be retrieved on time or when it is needed. Since the size of the disks is increasing and to remember the path of each and every file into the system is not an easy task. Desktop Search searches the database for the desired keyword, ranks it according to the similar content and then returns the required information with the best possible solution. Different types of search engines available are Crawler based search engines, Human powered directories, Meta search engines, and Hybrid search engines. Crawler based search engines create their listings automatically with the help of web crawlers. It uses a computer algorithm to rank all pages retrieved. These search engines are huge and retrieve a lot of information. It also contains some filters and ranking algorithm rank results in the best possible form.

Human powered directories are built by human selection i.e. they depend on humans to create (form) the repository. These directories can be organized or may be even unorganized depending upon the way by which they had been created, in desktop search where the files have been made by user and also the root and the sub root and sub directories as well, then the crawler based search engines might not be able to perform that well and also will take a lot of time to, and as generally too the crawler based searching is way too slow, to overcome these issues the crawler based search is infused with the spider search algorithm which is a more faster and reliable searching algorithm. Spider algorithm enlist the root directories present in the current search space and then crawls the files in those directories, it forms a tree like structure which helps in indexing files and folders structure of the current root folder and its subsidiaries too. Three main parts of a parser based search engine are Parser, Indexer and Searcher. The parser follows documents across the hard disk collecting information from different documents. Starting from a basic drive on the desktop, and recursively follow all the files and folders to other documents. This makes it possible to reach most of the hard drive in a relatively short time. The indexer takes the web pages collected by the parser and parses them into a highly efficient index. In the index, the terms are given an importance weighting by the search engine’s ranking algorithm. The searcher (query engine or retrieval engine) returns results of a query to the user. Because of these different term weighting and document selection methods, bias is introduced in the search engines. Different search engines also have different ranking algorithms and apply run-time filters to their results.

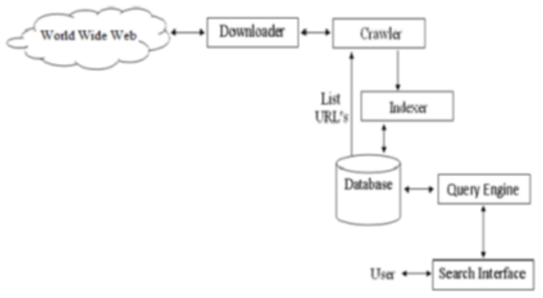


Fig. 1. Architecture of a typical web search engine

Desktop search engine the crawler is given a hard disk of the user from which it can extract documents one by one, parse and analyze these pages. On the other hand it provides a handle to the user to add a drive for extraction. It also extracts all the embedded URLs found in the page and add them to the list of URLs to be extracted. It uses in-place updating to maintain freshness of the database. A good indexing technique is used for searching the database with minimum possible time. The main working of base work search engine is shown in Figure 2 using a ‘0 level’ data flow diagram. The user submits query to the search engine and it searches for that query in the database of the crawler, and displays the result.

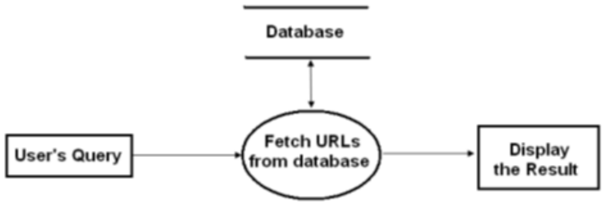


Fig. 2. 0 level DFD of proposed desktop search engine

The working of User operations and Admin operations is shown in Figure 3 using ‘1 level’ data flow diagram. In Admin Operation it shows scanning hard disk for files, parsing the text files from the drive and updating database. In User operations it shows submitting the Query and output operations. It also provides a handle to the user to select a hard disk for scanning as well as parsing the text files which are already there in the database of the search engine. The working of the crawler for base search engine is shown in Fig. 4. It starts with the base drive of the computer i.e. C Drive. First, a document is fetched from list of documents and checks for availability of that document in the database. If document is in the database then it uses the old page_id otherwise creates a new page_id. Now, the document is parsed for the text. The titles are inserted in the page table and the words of the page are extracted and placed in word table. If word is already in word table use the same word_id otherwise create a newword_id. Place the occurrences of the words in the occurrence table till all the words present in the file table are registered. These words are used in ranking. Fetch another document from FILE table. If the link already present in file table, then delete it from FILE table otherwise more links from FILE table are extracted if more links are present then they are extracted and whole process is repeated otherwise the crawling process is stopped.

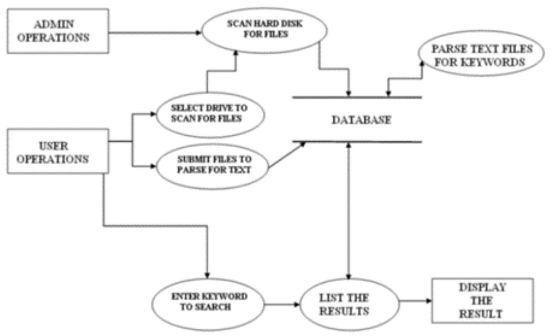


Fig.3. DFD representing administration operations and user's operations

4. Proposed methodology

A. *Filename fetch*

The first function fetches the filename and extension of the filename, to mark the search space attributes properly. Though to reduce the user complexity file search algorithm will still be operational without the extension keyword.

B. *Extension flag*

Algorithm for searching files includes a flag extension module which lets the algorithm decide whether to search with or without extension. Timings difference is always there in both the cases.

C. *Search algorithm*

The search algorithm used here can be termed as a Recursive File and Folder Search algorithm. The system initially reads the system location and then defines a home directory for itself as to return back from the search space folder to home folder. The System then forms a directory list of each folder and then separates out the files and the folders and then again enters a sub directory if needed or if there is any sub folder directory exists inside the root folder. The algorithm then takes a look back at the directory list and scans each string and matches the search query with the string list formed.

Algorithm: The algorithm coding steps are mentioned in this part:

- First step is to clear out the garbage values and clear the command window, history and the workspace.
- Step two is to determine the root folder for the algorithm or for the system, user dependent variable.
- Step three is to look for the search type, with extension or without extension.
- Step four is to take the user input for the filename.
- Step five is to take the user input for the filename extension.
- Step 6 is to check the system that whether the filename entered is in valid form or not empty.
- Step 7 is to check the system that whether the extension entered is in valid form or not empty.
- Store the current time of the system for timing calculations.
- Step nine is to start the timing internal clock for precise start to end timing calculations.
- Step ten is to store the directory files string into a variable "strt".
- Step eleven is to determine the size of the search space.
- Step 12 now runs a loop starting from 4 to i; where 'i' is size of the dir, though the size of the dir means the length of the root folder.
- Step 13 deals with the loop iteration for each string length upto the last point i.
- Loop iteration success and failure algorithm gets inserted into the string match algorithm as soon as the exact string matches it should return the file; else it should continue the loop iteration in the same folder.

It is very important to notice and analyze that the match string algorithm will get developed over the time only as it may include the some similar kind of string match such as 'town' and 'place' somewhat represents or interpret the same meaning hence the similar group of strings can be merge into the single group, another instance can be given of 'eat' and 'cat' they both have different meanings but their spelling is only differ by the first character so these can also be combined, considering the error percentage of an average human in typing is to be around 4%.

5. Results

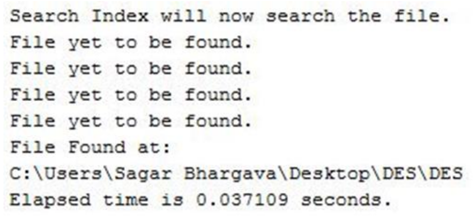
Table 1
Algorithm and time taken

Algorithm	With File Extension	Without File Extension
Recursive	0.12 seconds	0.03 seconds
Google	0.38 seconds	0.59 seconds
Yahoo	0.58 seconds	1.09 seconds

The algorithm shows different aspects of the desktop data searching and hence provides better searching for the user.

The work has been done over the timing constraint and also over the reducing complexities for the user in search criteria. Also it helps in mapping down all the system files separately in a different workspace and allows the algorithm to keep the track record of the data saved in the system.

The search engines listed above are available only for the online searching.



```
Search Index will now search the file.
File yet to be found.
File yet to be found.
File yet to be found.
File yet to be found.
File Found at:
C:\Users\Sagar Bhargava\Desktop\DES\DES
Elapsed time is 0.037109 seconds.
```

Fig. 4. Result section

However the recursive algorithm is available for the offline search and with change in platform it can be implemented over the online platforms as well.

6. Conclusion

Here from this paper it can be concluded that the timings has been gone down for the search results but with migrating the algorithm to a different platform such as Python, JavaScript or Perl it can be implemented for the online search space and can be proved out to be critical in terms of the search results timings.

References

[1] O. Bergman, R. Byth-Marom, R. Nachmias and S. Whittaker, Improved Search Engines and Navigation Preference in Personal Information Management, ACM Transactions on Information Systems, vol. 26, issue 4, (2008).
[2] C. Borjigin, Y. Zhang, C. Xing, C. Lan and J. Zhang, Dataspace and its Application in Digital Libraries, The Electronic Library, vol. 31, issue 6, pp. 688–702, (2013).
[3] M. Burghardu, T. Scheidermeier and Chtistian Wolff, Usability Guidelines for Desktop Search Engines, Proceedings of 15th International Conference on Human-Computer Interaction, Springer, pp. 176–183, LNCS 8004, (2013).
[4] Y. Cai, X. L. Dong, A. Halevy, J. M. Liu and J. Madhavan, Personal Information Management with SEMEX, Proceedings of International Conference on Management of Data, ACM SIGMOD, pp. 921–923, (2005).
[5] H. D. Chau, B. Myers and A Faulring, What to do When Search Fails: Finding Information by Association, Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp. 999–1008, (2008).
[6] J. Chen, H. Guo, W. Wu and C. Xie, Search Your Memory! Associative Memory Based Desktop Search System, Proceedings of the International Conference on Management of Data, ACM SIGMOD, pp. 1099–1102, (2009).
[7] B. Cole, Search Engines Tackles the Desktop, IEEE, (2005).
[8] http://www.copernic.com/en/products/desktopsearch/home/download.html
[9] E. Cutrell, D. C. Robbins, S. T. Dumais and R. Sarin, Fast, Flexible Filtering with Phlat-Personal Search and Organization Made Easy, Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp. 261–271, (2006).
[10] J. P. Dittrich, L. Blunschi, M. Farber, O. R. Giradm, S. K. Karakashian, M. Antonio and V. Salles, From Personal Desktops to Personal Dataspaces: A Report on Building the iMeMex Personal Dataspace Management System, Proceeding of BTW, pp. 292–308, (2007).
[11] J. P. Dittrich, iMeMex: A Platform for Personal Dataspace Management, Proceedings of 2nd NSF sponsored Workshop on Personal Information Management, ACM SIGIR, (2006).
[12] J. P. Dittrich and M. A. V. Salles, idm: A Unified and Versatile Data Model for Personal Dataspace Management, Proceedings of 32nd International Conference on Very Large Databases, pp. 367–378, (2006).
[13] D. Florescu, D. Kossman and I. Manolescu I, Integrating Keyword Search into XML Query Processing, Proceedings of International World Wide Web Conference, pp. 119–135, (2000).
[14] C. Hedeler, K. Belhajjame, N. W. Paton, A. Campi, A. A. A. Ferandes and S. M. Embury, Chapter-7 Dataspaces, Search Computing, Berlin Heidelberg Springer, pp. 114–134, LNCS 5950, (201).
[15] M. Kayest and S K Jain, A Proposal for Searching Desktop Data, Proceedings of 3rd International Conference on Innovations in Computer Science and Engineering (ICICSE), Springer, vol. 413, pp. 113–118, (2015).
[16] B. Markscheffel, D. Buttner and D. Fishcher, Desktop Search Engines A State of Art Comparison, Proceedings of the 6th International Conference on Internet Technology and Secure Transactions, pp. 707–711, (2011).
[17] S. Pradhan, An Algebraic Query Model for Effective Retrieval of XML Fragment, Proceedings of the 32nd International Conference on Very Large Databases, pp. 295–306, (2006).
[18] S. Pradhan, Towards a Novel Desktop Search Technique, Proceedings of 18th International Conference on Database and Expert Systems Applications, pp. 192–201, LNCS 4653, (2007).
[19] D. R. Virgilio, A. Maccioni and R. Torlono, A Unified Framework for Flexible Query answering over Heterogeneous Data Sources, Proceedings of 11th International Conference on Flexible Query Answering System, vol. 400, pp. 283–294, (2015).
[20] http://www.microsoft.com/windows/products/winfamily/desktopsearch default.mspx
[21] http://www.x1.com
[22] http://info.yahoo.com/privacy/in/yahoo/desktopsearch/, last visited on 10 January (2016).