Fuzzy File Management

Niloy Gupta
Computer Science & Engineering
National Institute of Technology
Karnataka, Surathkal, India
niloy_gupta@ieee.org

Abhinav K.R.
Computer Science & Engineering
National Institute of Technology
Karnataka, Surathkal, India
abhinav_kr@ieee.org

Annappa
Computer Science & Engineering
National Institute of Technology
Karnataka, Surathkal, India
annappa@ieee.org

Abstract—**In this paper we discuss a fuzzy logic approach to improve file management and organization. Incorporating fuzzy techniques in the file management process provides an intelligent way of maintaining files on the computer system. The fuzzy inference engine implements the decision making process required to select files for various file operations. This provides user convenience and system efficiency. We develop a fuzzy copy command in order to integrate fuzzy logic into the UNIX system's file management procedures and present examples which demonstrate the effectiveness of the command in organizing files.**

Keywords- Fuzzy Logic; Intelligent File Management; Intelligent File Organization, Fuzzy Operating System; UNIX File Organization.

I. INTRODUCTION

Effective file organization involves easy retrieval of files for reading, writing, updating and for performing file operations. It is a form of decision making where the user organizes or groups the files based on certain similarity factors. Creation of directories, deleting, copying and moving and grouping of files involves a decision making process [2,4]. Usually, users require non-precise file management techniques of copying or deleting files based on a particular relationship. This process generally involves retrieval of files from various directories and subsequent file operations on them [3].
Fuzzy logic can provide an intelligent file management technique which performs the decision making required in choosing the files on which operation is to be performed [1].

The paper is divided into four major sections. Section II discusses the fuzzification of file parameters. Section III describes the fuzzy rules required in decision making. Section IV investigates the application in the form of a fuzzy copy command for the Unix system. In Section V, we conclude with our general observations, inferences and applications.

II. FUZZY SIMILARITY MEASURES

Two files can be compared based on multiple features [9]. These features selected are based on file categorization techniques employed in personal information management [2,8]. We discuss the parameters involved in file comparisons and present a way of fuzzifying them. Before we fuzzify the file parameters, we define a degree or measure of similarity between two files for every file parameter. These, similarity measures (represented by the similarity coefficient β, $0 \le \beta \le 1$), indicate how closely the files are related [1,3]. $\beta_p=1$, implies that the file have the same value for the parameter p, and hence are same. A $\beta_p=0.8$, implies that the file are closely related to each other based on the parameter p.

A $\beta_p=0.1$, implies that the file are very different from each other based on the parameter p.

The similarity measures discussed below have been designed to convert the file parameters for the two files into metrics ranging from (0-1) indicating similarity measures.

A. *Filename*

Two files with similar filenames tend to be similar [2,7]. By convention, files related to a topic are named with a common pattern. E.g. assignment1.txt, assignment2.txt, song.mp3, song_lyrics.lrc. Thus, considering the above feature, the filenames of two files can be compared and a similarity index can be calculated.

$$\beta_{filename} = \frac{\chi(f_1,f_2)}{\text{minimum filename length of } (f_1,f_2)}$$

where $\chi(f_1,f_2)$ is the length of longest common substring between the files Hence, for the case of assignment1.txt, assignment2.txt; $\beta_{filename} = 10/11=0.909$

B. *Content*

Content of text files can prove to be a major factor in deciding the similarity between files [4,7]. Two files having the same content have a great degree of similarity and often the same file operation will be conducted on one will be conducted on the other. The similarity index of content

$$\beta_{content} = \frac{\omega(f_1,f_2)}{\text{minimum number of lines of } (f_1,f_2)}$$

where ω (f_1,f_2) returns the number of lines common to both the files

C. *Format*

Files with similar extensions or format share some degree of similarity. .txt and .doc formats have a greater similarity index when compared to .txt and .mp3. Thus, a table of similarity indexes is created between various file

TABLE I. Similarity indexes based on file formats

β_{format}	.txt	.doc	.mp3	.cpp

978-1-4244-8679-3/11/$26.00 ©2011 IEEE

.txt	1.00	0.90	0.35	0.80
.pdf	0.70	0.80	0.35	0.65
.java	0.75	0.70	0.40	0.80

Values based on experimental studies

formats and is referenced during decision making. Table I shows the similarity indexes between some file formats

D. Last Modification Date

The degree of similarity between files depends on a small extent on the time the files were last modified or created [7]. The similarity index of date

$$\beta_{date} = 1 - \frac{|y_1 - y_2|}{|y_1 + y_2|} - \frac{|m_1 - m_2|}{|m_1 + m_2|} - \frac{|d_1 - d_2|}{|d_1 + d_2|}$$

where y, m, d are the year, month and day of the last modification date of file f_1 and f_2 respectively.

E. File Size

The size of a file contributes to the degree of similarity between files. Groups of files in a particular directory having the same size, share to some extent some common relationship, based on which they can be grouped for file operation.

$$\beta_{size} = 1 - \frac{|x_1 - x_2|}{|x_1 + x_2|}$$

where x_1 x_2 are the file sizes of f_1 and f_2 respectively.

Given the above similarity measures we define fuzzy profiles for each of these parameters (viz. $\beta_{filename}$, $\beta_{content}$, β_{format}, β_{date}, β_{size}) by assigning memberships to their respective values.

The graph shown in Fig. 2 depicts the relationship. The similarity measures are fuzzified based in the fuzzy profiles.

The values are converted to fuzzy linguistic terms of low (0.0 -0.4), medium (0.3-0.7) and high (0.6-1.0) which indicate the level of similarity. This assists us in devising fuzzy rules that can map the user's desirability in choosing files for the file operation. Fig. 2 depicts the fuzzy membership function.

III. Fuzzy Decision Making

In order to perform the files operation on a group of files, the files must be clustered based on the overall degree of similarity. To do so, each file in a given search space must be compared to all other files based on the five parameters discussed above. After the similarity measures have been obtained and each measure has been fuzzified, the fuzzy rules are applied to infer the overall similarity between the files.

We employ the Mamdani Inference method [5,6,10] in deciding whether two files satisfy the similarity condition required by the file operation.

Our fuzzy rules can be divided into two cases, one in which both the files under comparison are regular (ordinary) files and the other where they are not regular files.

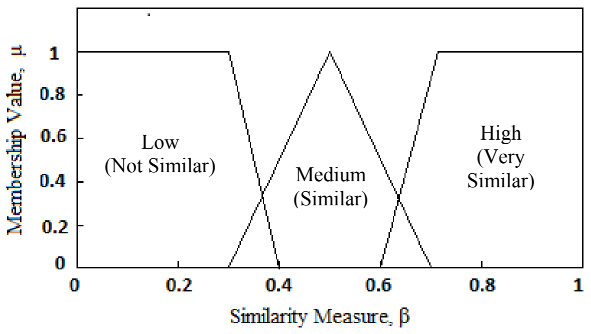

Figure 2. The fuzzy membership functions for similarity index β

Ordinary or regular files are of two types: 1. Text Files 2.Binary Files [11]. In the Unix system, content comparisons can be made only on regular files. Thus, the content similarity measure $\beta_{content}$ plays a role in the fuzzy decision making only in the case of ordinary files. It should also be known that $\beta_{content}$ contributes significantly to the total similarity measure between the files. Two files with the same content will have a high similarity measure irrespective of the filename, format or date of modification.

However, when content cannot be compared as in the case of media files, the filename plays a crucial role in determining the file grouping decision.

Table II and Table III depict some of the rules used in the fuzzy inference process. These fuzzy rules have been devised based on investigative study on the file management habits and techniques used by ordinary users [1,2].

After the application of fuzzy rules, we need to defuzzify the results. We implement the centroid method [5,6,10] to obtain the crisp overall similarity measure β_{final}. β_{final} is the deciding factor based upon which the file operation is performed.

The user is given flexibility in deciding how strict the file grouping should be. Four thresholds have been provided for the user, T_1 very strict (0.8-0.9), T_2 moderately strict (0.6-0.9), T_3 less strict (0.5-0.7), T_4 not strict (0.0-0.5). Depending upon the user's choice of strictness, if β_{final} falls above the specified threshold, the file operation is performed else the process is abandoned.

Thus, summarizing the entire process, each file in the search or directory space is compared with the other files based on the five similarity measures discussed above. If they are regular files, the $\beta_{content}$ is considered, else left out. After the fuzzification, application of fuzzy rules and defuzzification, the final result β_{final} is obtained. If β_{final} lies in the range of the specified strictness, the file operation (copy, delete move, search etc) is performed.

IV. Unix Inplementation

We present a UNIX implementation of the fuzzy file management procedure.

The filename and date of modification can be obtained using the "ls -la" terminal command [11]. The filenames, year, month and date can be extracted by piping it to "cut –f", while the size of the file, in bytes, can be obtained using the

TABLE II. Some of the fuzzy rules used in the comparison of regular files

$\beta_{content}$	$\beta_{filename}$	β_{format}	β_{date}	β_{size}	β_{final}
High	High	High	High	High	Very similar
High	High	Medium	Medium	Medium	Similar
High	High	Medium	Low	Low	Similar
High	Low	Medium	Medium	Medium	Similar
Low	Medium	High	High	Medium	Not Similar
Medium	High	High	High	High	Very Similar
Medium	Medium	Medium	Medium	Medium	Similar
Low	Low	Medium	Medium	Medium	Not Similar
Low	High	High	High	Medium	Similar

TABLE III. Some of the fuzzy rules used in the comparison of non-regular files

$\beta_{filename}$	β_{format}	β_{date}	β_{size}	β_{final}
High	High	High	High	Very Similar
High	Medium	Medium	Medium	Similar
High	Medium	Low	Low	Similar
Low	Medium	Medium	Medium	Not Similar
Medium	High	High	Medium	Similar
High	Low	High	High	Very Similar
Medium	Medium	Low	Low	Similar
Low	High	Medium	Medium	Similar
High	High	High	Medium	Very Similar
Low	Low	Low	Low	Not Similar

"du -b" command. Therefore, βdate, βfilename and βsize can be conveniently determined. For extracting the length of the longest common substring between the filenames a simple string handling function can made which returns χ(f1,f2). The extensions of filenames can be extracted using "${f1 #.}" where f1 is the filename.

The content of two sorted files can be compared using the "comm -12" command, [11] which returns the lines common to both the files ω (f1, f2). The "wc" command can be piped with it obtain the number of common lines. Also the "sort" command [11] can be used to sort the two files. The "wc" command [11] is also used to obtain the number of lines of the regular file. These help in determining βcontent.

Thus, a simple bash script or system file can be written which can perform all the computation of similarity indexes and fuzzy values maintain a table of the computed results for every directory and perform the file operation. The table is updated whenever the file or directory is modified or updated. The purpose of maintaining a table of fuzzy similarity values saves the cost of computing the values again. For n number of files in the directory n C 2 comparisons are made every time the fuzzy file operation is performed. As the number of files increases, the computation time increases. Therefore, the complexity of the entire fuzzy file management procedure will be of the order Θ(n2), where n is the number of files in the directory search space. Thus, maintaining a list of pre-computed values saves the time and processor cost of re-computation at the expense of small amount of memory.

The entire process can be integrated into a UNIX file management command. The format of the command is as follows:

FuzzyCommand <Subject> <source directory path> [destination directory path] <Threshold>

The threshold argument is the linguistic terms mentioned before, very strict, moderately strict, less strict, not strict. The subject argument requires the user to specify what kind of files the user wishes to target. The arguments can be of the form of Audio, Text, Video, Scripts, and Null etc depending upon the implementation. This reduces the amount of files to be compared and provides a more accurate method of selecting files for the operation. If the subject is given as Null then all files will be compared and the content parameter will be considered only for comparison between regular files.

Case Study: Fuzzy Copy

Consider a fuzzy copy command of the following format: fuzzycopy <subject> <source directory path> <destination directory path> <Threshold>

Example 1:

Let the source folder contain the following files: list1.txt, list2.txt, test.txt. Where the contents of list1 and list2 are the same and test has no common data with list1 and list2. Also, all the three files have been created on the same day and have the same size of 50KB.

The user has given the argument of subject as text files and that of the threshold as very strict. Consider the comparison between list.txt1 and list2.txt. βfilename = 0.75, βcontent=1, βformat=1, βdate=1, βsize=1

By applying the fuzzy rules and after defuzzification we obtain a final similarity measure βfinal =0.82, which is in the region of very strict and hence the files are copied to the destination folder.

Now, the comparison of list1.txt/list2.txt with test.txt yields βfilename = 0.5, βcontent=0, βformat=1, βdate=1, βsize=1. After the application of the entire process we get βfinal = 0.189 which is below the threshold and hence the file is not copied.

Example 2:

Let the two files in the directory be 1. Helloworld.mp3 2.Helloworldlyrics.txt. The subject is Null and threshold is less strict.

$\beta_{filename} = 0.6$, $\beta_{format}=0.35$, $\beta_{date}=1$, $\beta_{size}\approx 0$

The fuzzy implementation yields $\beta_{final}=0.5$. Thus, the audio file and its lyrics get copied.

V. CONCLUSION

Through analysis and implementation of fuzzy techniques in file management and organization, we have found that it provides an intelligent way of performing file operations on a large group of files without the need for the user to locate and decide upon the files.

It should be noted that there were cases when the file desired by a particular user were not selected or grouped. One of the methods to overcome this drawback is to develop a learning system [3], which can learn the user's file management habits and alter the fuzzy rules or thresholds.

By an overall perspective, it can be concluded that incorporating fuzzy logic into file process provides user convenience and system efficiency.

ACKNOWLEDGMENT

The authors would like to thank Mr. Tejovanth N and Mr. Aniruddh Galgali, Department of Electrical and Electronics Engineering, National Institute of Technology Karnataka, for their valuable advice which has greatly improved the quality of the paper.

REFERENCES

[1] Abraham Kandel, Yan-Qing Zhang, Marlow Henne "On use of fuzzy logic technology in operating systems," Fuzzy Sets and Systems 99 pp. 241-251, November 1996.

[2] Lansdale, M., "The Psychology of Personal Information Management", Applied Ergonomics, 1988, 19(1), 55–66.

[3] M. Dash & H. Liu "Similarity Detection among Data Files - A Machine Learning Approach",IEEE Knowledge and Data Engineering Exchange Workshop, 1997, pp. 172-179.

[4] Ali Sajedi Badashian, Seyyed Hamidreza Afzali, Morteza Ashurzad Delcheh, Mehregan Mahdavi, Mahdi Alipour, "Conceptual File Management: Revising the Structure of Classification based Information Retrieval", Proc. Fifth International Conference on Digital Information Management(ICIDM 10), IEEE press, July 2010, pp. 108-113, doi:10.1109/ICDIM.2010.5664670

[5] Ebrahim H. Mamdani, "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis", IEEE Transactions on Computers, vol. 26, pp. 1182-1191, December 1977.

[6] Mamdani E.H., Assilian S., "An experiment in linguistic synthesis with a fuzzy logic controller", International Journal of Man-Machine Studies, Vol.7, No. 1, 1975.

[7] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R. and Robbins, D. C. "Stuff i've seen: a system for personal information retrieval and reuse". In SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, New York, NY, USA, 2003. ACM Press, 72–79.

[8] Barreau, D.and Nardi, B., "Finding and Reminding: File Organization from the Desktop", SIGCHI Bulletin, 1995, 27(3), 39–43.

[9] Silberschatz, A., Peterson, J. L., and Galvin, P.B.2006.Operating System Concepts.7th Edition Addison.

[10] Elaine Rich, Kevin Knight, Shivashankar B Nair, P.B.2010.Artificial Intelligence. Third Edition, pp. 445-455

[11] GNU Bash Reference Manual (Online) , http://www.gnu.org/ software/bash/manual/bashref.html