Artif Intell Rev (2009) 31:101–117
DOI 10.1007/s10462-009-9129-2

Managing computer files via artificial intelligence approaches

Xiaolong Jin · Jianmin Jiang · Geyong Min

Published online: 24 October 2009

Abstract Agent-oriented computing has been regarded as a very promising methodology to developing intelligent software systems. Intelligent agent technology has, thus, been successfully applied in many industrial and commercial areas. Cased based reasoning (CBR) is an effective and efficient analogical reasoning method for solving problems using the knowledge of past experiences, which are stored in a knowledge base as cases. CBR has been extensively employed to tackle such problems as design, planning, classification, and advising in many different application fields. On the other hand, as various files are created on computers, how to efficiently manage computer files becomes a significant issue. So far, there are a number of file management systems available. However, none of them can deal with these crucial problems of file management: Which files should be deleted after their use? Which files should be temporarily kept or permanently preserved? To the best of our knowledge, these problems have not yet been investigated in the open literature. To bridge this gap, in this paper we explore the value of the above artificial intelligence approaches in managing computer files. We develop an intelligent agent based personal file management system, where CBR is employed to guide users to managing their files. Through extensive practical experiments, we validate the effectiveness and efficiency of the developed system.

Keywords Intelligent agents · Multi-agent systems · Case based reasoning · Similarity measurement · File management

X. Jin (✉) · J. Jiang · G. Min
Digital Media and Systems Research Institute, School of Computing, Informatics and Media,
University of Bradford, Bradford BD7 1DP, UK
e-mail: x.jin@brad.ac.uk

J. Jiang
e-mail: j.jiang1@brad.ac.uk

G. Min
e-mail: g.min@brad.ac.uk

1 Introduction

Agent-oriented computing has been extensively regarded as a very promising methodology to developing intelligent software systems, as this computing paradigm enables software engineers to model applications in a natural way that resembles how humans perceive the problem domains (Braubach et al. 2003; Chmiel et al. 2005; Jennings 2001). Particularly, it has been shown that this computing paradigm is well suited to handle the complexity of developing software in modern application scenarios (Zambonelli and Omicini 2004). Therefore, Intelligent Agent Technology (IAT) has been successfully applied in many industrial and commercial areas, including information retrieval and filtering, electronic commerce, human computer interaction, telecommunication systems, air traffic control, planning and scheduling, process control, manufacturing, workforce management, and military (Luck et al. 2005).

When people face new problems, they often derive solutions based on their past experiences with similar situations. Cased based reasoning (CBR) is exactly originated from this problem solving pattern (Craw et al. 2006; Watson and Marir 1994). CBR implicitly hypothesizes that "similar problems have similar solutions". It is an effective analogical reasoning method for solving problems using the knowledge of past experiences, which are stored in a knowledge base as cases (Watson and Marir 1994). CBR is particularly useful in those poorly understood or dynamically evolving domains, where knowledge is difficult to be formalized (Glasgow et al. 2006). CBR has been widely adopted to solve the problems of design, planning, classification, and advising in many different application fields, such as, image processing (Khanum and Shafiq 2006; Perner 2002), molecular biology (Glasgow et al. 2006), course timetabling (Burke et al. 2006), spam filtering (Delany and Bridge 2006), and fault diagnosing (Iglesias et al. 2008; Pous et al. 2008; Ross et al. 2002).

Since the great invention, computers (e.g., desktops, laptops, PDAs) have gradually entered people's daily life and have already fundamentally changed the ways in which people live, work, and play. Every day, various new files are created and stored on each computer. As a consequence, two significant problems emerge: first, since there are too many electronic files stored on computers, if they are not well managed, it will be very difficult for computer users to find and use their desired files; second, the large number of computer files require substantive storage space. However, although the capacity of consumer hard disks has been increased from a few megabytes in 1980s to up to several thousand gigabytes nowadays,[1] the storage space is still regarded as a scare resource. Therefore, how to efficiently manage electronic files stored on computers so as to not only facilitate human users to use them, but also save storage space is an important issue.

Thus far, besides the file management systems embedded in the commonly used operating systems (e.g., Explore in Microsoft Windows), a lot of stand-alone file management systems have been made available (Carlton 2005; Eder et al. 2000; Mahalingam et al. 2003; Wedde et al. 1990; You et al. 2005). However, unfortunately, there are no file management systems that are able to automatically deal with the following crucial problems of managing computer files:

(1) Which files shall be deleted from computers after their use?
(2) Which files need to be temporarily remained for a period of time?
(3) Which files should be preserved permanently?

To fill this gap, this study intends to explore the value of artificial intelligence in managing computer files. Specifically, we develop an intelligent agent based personalized file

[1] http://www.pcguide.com/ref/hdd/histTrends-c.html.

management system, which can automatically deal with the above problems by integrating the IAT and CBR technologies. In this management system, intelligent software agents cooperate and coordinate with each other to manage computer files such that they are deleted, temporarily remained, or permanently preserved. As the key component of the system, a CBR-based recommendation agent is engineered to recommend suitable actions on individual files using the knowledge learned from the behaviors of human users.

The rest of the paper is organized as follows: In Sect. 2, the preliminaries and related work are briefly introduced. Section 3 presents the architecture of the multi-agent based file management system and describes the functionality of individual agents. In Sect. 4, we discuss the representation of cases and the similarity measurements on the attributes of different cases. Section 5 presents the CBR recommendation agent and the case base management agent, as well as their CBR mechanism. Specifically, the important issues of cases, namely, retrieval, reuse, revision, retainment, and management, are investigated. We present our experimental results to validate the application of the CBR mechanism to file management in Sect. 6. Finally, Section 7 concludes the paper.

2 Preliminaries and related work

In this section, we briefly review the preliminaries and related work on intelligent agents and multi-agent systems, CBR, and file management systems.

2.1 Intelligent agents and multi-agent systems

In the agent-oriented computing paradigm, intelligent agents are the fundamental building blocks. An intelligent agent is an encapsulated software system that is situated in a dynamically changing environment and is capable of flexible, autonomous action in that environment in order to achieve its design objectives (Jennings 2000; Jennings et al. 1998). Particularly, an agent possesses the following three important characteristics (Jennings 2001; Lind 2000; Zambonelli and Omicini 2004):

– *Autonomy* An agent is not passively subject to a global, external flow of control in its actions. In other words, an agent has its own internal execution activity. It is pro-actively oriented to the achievement of a specific task.
– *Situatedness* An agent is always situated in a particular environment, no matter it is a computational or a physical one, and it is able to sense and affect (portions of) such an environment.
– *Sociality* Agents usually work in open operational environments hosting the execution of a multiplicity of agents, possibly belonging to different stakeholders.

In multi-agent systems, the global behavior derives from the interactions among the constituent agents. In fact, agents may communicate/coordinate/cooperate with each other (in a dynamic way and possibly according to high-level languages and protocols) either to achieve a common objective or because this is necessary for them to achieve their own objectives. Therefore, it is the interactions among agents that make intelligent agents and multi-agent systems a valuable metaphor in computing and make them attractive when they are adopted to tackle complex application scenarios. Besides, dividing functionality among many agents provides modularity, flexibility, modifiability, and extensibility.

Due to these appealing features, IAT has gained great success in many different application domains. For instance, Hellingrath et al. (2009) investigated the application of IAT to supply chain management. They analyzed the requirements on multi-agent systems used for the supply chain management domain and further proposed a framework for the development of such multi-agent systems. They designed and implemented common modules in order to provide both sophisticated and generic basis for specific projects. Yang et al. (2008) developed a reactive multi-agent system for vehicle collision avoidance, where individual agents interact with other agents and the obstacles situated in the environment using the so-called physics-inspired behaviors. Collision avoidance in such a multi-agent system emerges as the global result of the localized interactions among individual agents. In the age of information economy, dynamic business alliance is a natural phenomenon of fierce competition in economic globalization. Therefore, Sun et al. (2008) investigated the application of IAT in the construction of dynamic alliance. They developed a framework for partner selection, contract network based negotiating agreement, and negotiation process with resource constraints.

2.2 Case based reasoning

Generality speaking, a CBR system consists of a case base, a similarity function for measuring the similarity between a new problem at hand and an existed case, some rules for case retrieval and revision, and an algorithm for deriving solutions to new problems (Khoshgoftaar et al. 2006). In a CBR system, a new problem is solved by investigating, matching, and reusing solutions to similar cases that have been previously solved. At the highest level of generality, CBR can be formalized as a four-step procedure (see Fig. 1):

(1) *Retrieve* Given a new problem, retrieve cases from the case base, which are similar to this problem to a certain extent.
(2) *Reuse* Investigate the solutions of the retrieved similar cases to derive a solution to the new problem.
(3) *Revise* Test the solution to the new problem and revise it, if necessary, using predefined revision rules or heuristics.
(4) *Retain* After the solution has been successfully adapted to the new problem, decide whether or not to store the new problem and its solution into the case base as a new case.

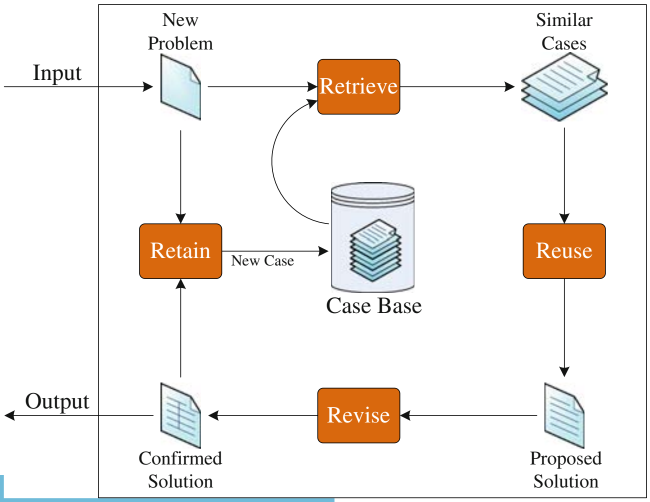

Fig. 1 A schematic diagram of a Cased Based Reasoning system and its four-step procedure

By retaining new cases, a CBR system is able to learn new knowledge throughout its life cycle (Burke et al. 2006).

Similar to IAT, CBR has also been successfully applied into many application fields. For example, Burke et al. (2006) developed a CBR system for scheduling educational courses, where past high-quality timetables are stored in a case base to help produce future timetables efficiently. In this system, the case base is organized as a decision tree. The retrieval process chooses those cases that are graph isomorphic to the new case in their sub-attributes. Spam filtering is a challenging issue for email server systems. CBR was employed to filter unsolicited bulk messages in Delany and Bridge (2006); Delany et al. (2005). Specifically, the authors developed a system called E-mail Classification Using Examples (ECUE), where two CBR-based alternatives, feature-based and feature-free, are invented. As determining the three-dimensional structure of a protein is an important step towards understanding its biological function, Glasgow et al. (2006) developed a CBR system to predict the structures of news proteins from their contact maps based on the detailed knowledge of the chemical and physical properties of proteins. CBR was also adopted for the software fault prediction problem in Khoshgoftaar et al. (2006). In more detail, a CBR system was developed as a software fault prediction model to quantify the expected number of faults in a module under development based on similar modules that have been previously developed.

2.3 File management systems

As we have mentioned previously, so far lots of stand-alone file management systems have been made available in the open literature for either general or specialized purposes (Carlton 2005; Eder et al. 2000; Mahalingam et al. 2003; Wedde et al. 1990; You et al. 2005). For instance, a multimedia document filing system was invented to support more effective conceptual-based or content-based document retrieval in Fan et al. (1997). In this system, a document type hierarchy model is developed to identify document types by analyzing layout, conceptual, and content structures. A user-defined folder organization is employed to store frame instances and the high level conceptual information of documents. Finally, a storage architecture is presented to integrate the document type hierarchy model, the folder organization, and the original document storage as a three level storage system. In order to assure that deleted files are completely erased from storage devices, Joukov and Zadok (2005) designed a file system extension, called Purgefs, which transparently overwrites files on the per-deletion basis. A personal file management system was developed, which can provide transparent and reliable access to personal files distributed on different file servers in Mutka and Ni (1992). Besides, as recognizing that the simple hierarchical name space of existing file systems is not efficient in managing nowadays computer files that have rich semantics, Mahalingam et al. (2003) invented and implemented a novel archival file system, namely, Sedar, which can store, manage, and retrieve computer files in a semantic manner. Unfortunately, to the best of our knowledge, none of the available file management systems can deal with the crucial problems mentioned in the previous section.

3 A multi-agent system for managing computer files

In this paper we refer file management to the decision making on whether to delete, keep, or preserve computer files. We will not consider such issues as how to efficiently classify and organize files according to their specific contents, where and how to store files such that

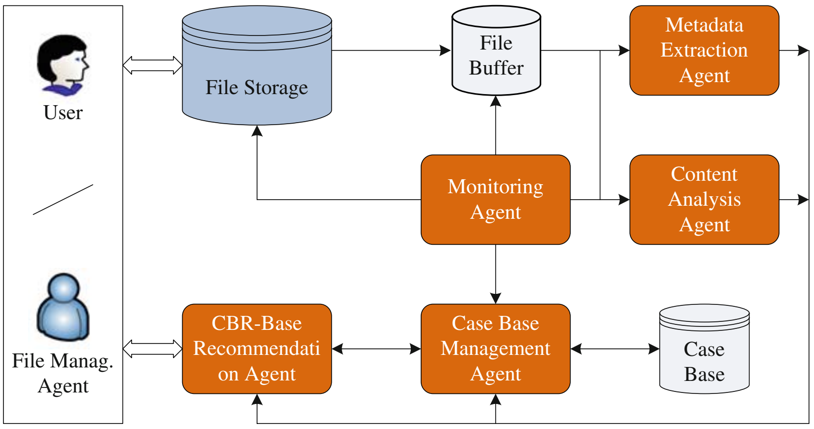

Fig. 2 A schematic diagram of the multi-agent based personal file management system

they can be readily accessed. All of such issues are beyond the scope of this paper. In Fig. 2, we present a schematic diagram of the multi-agent system that we have developed to manage computer files. In what follows, we will briefly introduce the functionality of individual agents involved in this system.

3.1 Human user/file management agent

The human user or the file management agent is responsible for managing computer files stored in the storage space that can be a logical disc driver or a file directory. The human user or the file management agent can consult the CBR-based recommendation agent on whether a file at hand should be deleted, kept, or preserved. If the human user is operating the system, She/he will make the final decision. In this case, if the user's final decision is different from the recommendation made by the CBR recommendation agent, the multi-agent system will learn from this situation. However, if it is the file management agent that is responsible for the whole system, it will usually accept the recommendation made by the CBR recommendation agent.

3.2 Monitoring agent

This agent monitors the actions of the human user or the file management agent on computer files and notifies its final decisions (i.e., delete, keep, or preserve) on files to the case base management agent. The monitoring agent is also responsible for temporarily copying the files under consideration into the preassigned file buffer for the feature extraction agent and the content analysis agent, which will extract the features of individual files and analyze their contents, respectively. After the extraction, those files will be removed from the file buffer.

3.3 Metadata extraction agent

This agent is designed to extract detailed metadata (e.g., size, type, and creation time) of individual files that have been temporarily put into the file buffer. The extracted metadata will be used by the CBR recommendation agent as the foundation to make suitable recommendation.

3.4 Content analysis agent

The functionality of this agent is to extract and analyze the specific contents of computer files. At the present system, this agent works only on document files. It can extract their textual contents and lexically analyze them to obtain a list of keywords.

3.5 Case base management agent

A case base is provided to support the recommendation of the CBR mechanism, where the detailed information of deleted and preserved files is stored as past cases. Note that the files that are determined to be temporarily kept are not stored in the case base. The case base management agent is in charge of the case base. Specifically, it is responsible for storing new cases into the case base, retrieving existing cases based on certain criteria, and filtering cases whose performance cannot meet certain predefined requirements. Besides, this agent acts as the supporter of the CBR recommendation agent.

3.6 CBR recommendation agent

This agent employs the CBR mechanism to make suitable recommendation on deleting, keeping, or preserving computer files. It relies on the similar cases provided by the case base management agent and the new case to make suitable recommendation.

In Algorithm 1, we present the detailed procedure of the file management system for managing computer files.

4 Case representation and similarity measurements

In order to employ the CBR mechanism to manage computer files, how to represent cases and how to measure the similarity between the attributes of cases are two fundamental technical issues, which will be discussed in this section.

4.1 Case representation

Generally speaking, a case represents knowledge about a particular problem solving experience and hence includes a problem description, a solution to the problem, and the feedback, if available, on the success of the solution (Glasgow et al. 2006). More specifically, a case is usually represented by a list of attribute-value pairs that represent the values of different attributes of the original problem. In our CBR system, each case actually corresponds to a computer file that has been deleted or deemed for preservation. A case consists of three types of attributes, namely, *descriptive attributes*, *solution attribute*, and *performance attributes*. Figure 3 presents a complete list of attributes used for representing cases.

4.1.1 Descriptive attributes

The descriptive attributes of a case refer to those attributes that represent the detailed information, including metadata and contents, of the corresponding file. Figure 3 itemizes all descriptive attributes. Note that in this study the keyword set, kwd, of a case is defined as an

Algorithm 1 The procedure of the agent-based personal file management system for managing computer files

```
Require: Given a set of files, $\mathbb{F}$, the file management system is required to recommend suitable actions
    (delete, keep, or preserve) on individual files.
 1: for each file $f$ in $\mathbb{F}$ do
 2:   The monitoring agent copies $f$ to the file buffer;
 3:   The metadata extraction agent extracts all required metadata from $f$ and submits to the CBR-based
      recommendation agent and the case base management agent;
 4:   The content analysis agent analyzes the contents of $f$ to obtain its keywords, and finally submits to the
      CBR-based recommendation agent and the case base management agent;
 5:   The monitoring agent removes $f$ from the file buffer;
 6:   Taking the metadata and keywords of $f$ as a new case, the case base management agent retrieves a set,
      $\Phi$, of similar cases from the case base and submits to the CBR recommendation agent;
 7:   Based on the solutions of the similar cases, the CBR recommendation agent recommends an action on
      the new case to the human user or the file management agent. This step may be repeated, if necessary;
 8:   The human user or the file management agent makes the final decision on $f$ based on the recommendation
      from the recommendation agent and performs the selected action finally;
 9:   The monitoring agent monitors the final action on file $f$ and informs it to the CBR recommendation
      agent and the case base management agent;
10:   if the final decision is the same as its recommendation then
11:     The CBR recommendation agent simply discards the metadata and keywords of $f$;
12:   else
13:     The CBR recommendation agent integrates the metadata and keywords of $f$ and the final decision
          as a new case;
14:     The case base management agent stores the new case into the case base;
15:   end if
16:   for each case $F_X$ in $\Phi$ do
17:     Increase its recommendation counter by one;
18:     if the solution in $F_X$ is the same as the final decision of the human user or the file management agent
          on file $f$ then
19:       Increase its correct recommendation counter by one;
20:     else
21:       Increase its incorrect recommendation counter by one;
22:     end if
23:   end for
24: end for
```

ordered set. The higher the order of a keyword in kwd, the better it can be used to describe the contents of the case.

4.1.2 Solution attributes

Figure 3 shows that there are two attributes in a case for representing its solution. Specifically,

(1) Final action act records the final decision made by the human user or the file management agent on the corresponding file. As aforementioned, the final action attribute of a case has three possible values, namely, delete, keep, and preserve.
(2) Action time fat refers to the specific time when the final action is performed on the file.

4.1.3 Performance attributes

Unlike the above two types of attributes, the performance attributes of a case is a set of attributes that indicate the performance of this case on recommending suitable solutions to

Fig. 3 The descriptive attributes, solution attribute, and performance attributes for representing cases

Description Attributes:

- Type of the file, tpf;
- Size of the file, szf;
- Time related attributes, tra;
  - Creation time, crt;
  - Last access time, lat;
  - Last modification time, lmt;
- Binary descriptive attributes, tra;
  - Is a hidden file, ihf;
  - Is a read-only file, irf;
  - Is a system file, isf;
  - Is a temporary file, itf;
- Keywords of the file, kwd.

Solution Attributes:

- Final action, act;
- Action time, fat.

Performance Attributes:

- Total recommendations, trd;
- Correct recommendations, crd;
- Incorrect recommendations, ird.

new problems. In the file management system, there are three performance attributes for each case:

(1) Total recommendations trd is a counter that counts the total times when this case is retrieved as a similar one to the case at hand based on a predefined similarity function.
(2) Correct recommendations crd is also a counter for recording the times when the case successfully recommends a final solution to a new case. In other words, the recommended action for a file is the same as the final action taken by the human user or the file management agent.
(3) Incorrect recommendations ird counts the times when the case fails to predict the final solution of a new case. In other words, the recommended action for a file is different from the final action taken by the human user or the file management agent.

4.2 Similarity measurements

Given a file, f, in order to recommend a suitable action on it, the case base management agent will take it as a new case and then retrieve the n nearest neighboring cases from the case base, whose descriptive attributes are most similar to those of file f. In doing so, the key is to measure the similarity between the corresponding attributes of different cases. Note that for all similarity measurement we normalize them into a value in [0, 1]. The larger the similarity value, the more similar the two cases. Later on, we suppose F_A and F_B be two cases for measuring the similarity between their attributes and denote a certain attribute x of a case F_X as F_X^x.

4.2.1 Similarity on file types

We use a tree structure to organize various types of computer files. Figure 4 presents a schematic diagram of the tree structure. It can be noted that the highest level (i.e., the root) of

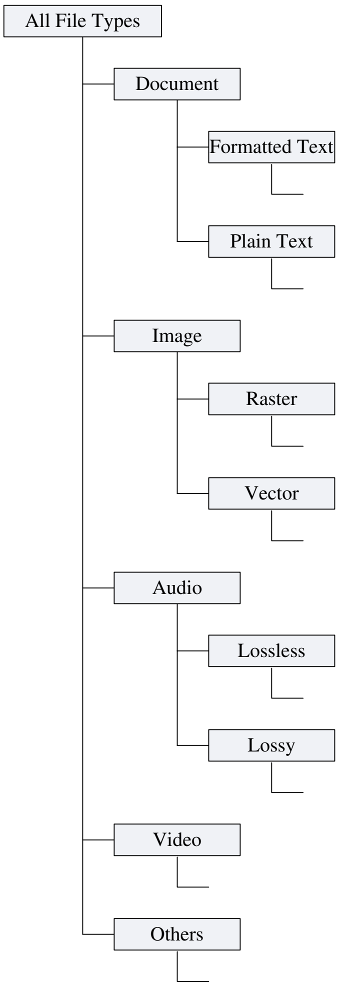

Fig. 4 A tree structure model of file types

the tree structure is the full set of all file types. At the second highest level, files are classified into five types, namely, *document*, *image*, *audio*, *video*, and *others*. Each type has several subtypes or a sub-tree structure. For instance, image files can be classified into two subtypes, namely, *raster format images* and *vector format images*. Further, raster images have different formats, such as, bmp, tif, tiff, jpeg, jpg, mng, and psd. Similarly, vector images have also a serious of formats, namely, svg, ps, wmf, swf, etc. Let F_A^{tpf} and F_B^{tpf} be the type attribute of cases F_A and F_B, respectively. F_A^{tpf} and F_B^{tpf} are two leaf nodes in the above tree structure of file types. Let π be the closest common ancestral node of F_A^{tpf} and F_B^{tpf}. Next, we can define the similarity F_A^{tpf} and F_B^{tpf} as follows:

– If π is a leaf node, then $F_A^{tpf} = F_B^{tpf} = \pi$. Hence, we have $\mathcal{S}(F_A^{tpf}, F_B^{tpf}) = 1$;
– If π is the root node, $\mathcal{S}(F_A^{tpf}, F_B^{tpf}) = 0$;
– Otherwise,

$$\mathcal{S}(F_A^{tpf}, F_B^{tpf}) = 2^{-(\max\{\text{step}(F_A^{tpf}),\text{step}(F_B^{tpf})\}-\text{step}(\pi))}, \quad (1)$$

where step(x) is a function for counting the steps from the root of the tree structure of file types to its offspring node, x.

4.2.2 Similarity on file sizes

Generally speaking, the sizes of files of different types are significantly different. For example, a video file is usually up to several hundred megabytes, while a document file is often about several hundred kilobytes. This fact should be considered in measuring the similarity of file sizes. Let F_A^{szf} and F_B^{szf} be the size attribute of cases F_A and F_B, respectively. We can define the corresponding similarity measure as follows:

$$\mathcal{S}(F_A^{szf}, F_B^{szf}) = \begin{cases} 1 - \frac{1}{\alpha} \cdot \left| F_A^{szf} - F_B^{szf} \right|, & \text{if } F_A^{tpf} = F_B^{tpf}, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\alpha = \max_{F_X \in \aleph} F_X^{szf} - \min_{F_Y \in \aleph} F_Y^{szf}$ and $\aleph$ is the set of cases in which we have $\forall F_Z$, $F_Z^{tpf} = F_A^{tpf} = F_B^{tpf}$.

4.2.3 Similarity on binary descriptive attributes

As we have noticed in the previous subsection, there are four binary descriptive attributes for each case, namely, $\{ihf, irf, isf, itf\}$. The similarities between these binary attributes of two cases can be readily measured. Let F_A^{bda} and F_B^{bda} be the values of a certain binary descriptive attribute of cases F_A and F_B, respectively. The corresponding similarity can be calculated as

$$\mathcal{S}\left(F_A^{bda}, F_B^{bda}\right) = 1 - \left| F_A^{bda} - F_B^{bda} \right|. \tag{3}$$

4.2.4 Similarity on time related descriptive attributes

In the case representation, we have three time related descriptive attributes, namely, $\{crt, lat, lmt\}$, to characterize cases. Let F_A^{tra} and F_B^{tra} be the values of a time related attribute corresponding to cases F_A and F_B, respectively. The similarity measure can be defined as follows:

$$\mathcal{S}\left(F_A^{tra}, F_B^{tra}\right) = 1 - \frac{1}{\beta} \cdot \left| \left(F_A^{fat} - F_A^{tra}\right) - \left(F_B^{fat} - F_B^{tra}\right) \right|, \tag{4}$$

where $\beta = \max_{F_X \in \aleph} \left(F_X^{fat} - F_X^{tra}\right) - \min_{F_Y \in \aleph} \left(F_Y^{fat} - F_Y^{tra}\right)$ and $\aleph$ denotes the case base.

4.2.5 Similarity on the keywords of files

Let $F_A^{kwd} = \{k_A^1, k_A^2, \ldots, k_A^{N_A}\}$ and $F_B^{kwd} = \{k_B^1, k_B^2, \ldots, k_B^{N_B}\}$ be the ordered sets of keywords corresponding to cases F_A and F_B, respectively, where N_A and N_B are the numbers of keywords of F_A and F_B. We measure the keywords similarity between F_A and F_B as follows:

$$\mathcal{S}(F_A^{kwd}, F_B^{kwd}) = \frac{1}{\max\{N_A, N_B\}} \times \sum_{\kappa \in F_A^{kwd} \cap F_B^{kwd}}^{\kappa = k_A^i = k_B^j} \left\{ 1 - \frac{|i-j|}{\max\{N_A, N_B\}} \right\}, \tag{5}$$

where κ is a common keyword in F_A^{kwd} and F_B^{kwd} and i and j are its orders in these two keyword sets, respectively. Obviously, Eq. 5 fully takes into account the orders of individual keywords in the corresponding keyword sets. Particularly, this similarity measure has the following properties:

– If $F_A^{kwd} = \emptyset$, $F_B^{kwd} = \emptyset$, or $F_A^{kwd} \cap F_B^{kwd} = \emptyset$, $\mathcal{S}\left(F_A^{kwd}, F_B^{kwd}\right)$ is minimized to zero.
– If $F_A^{kwd} = F_B^{kwd} \neq \emptyset$, $\mathcal{S}\left(F_A^{kwd}, F_B^{kwd}\right)$ is maximized to one.
– Otherwise, $\mathcal{S}\left(F_A^{kwd}, F_B^{kwd}\right) \in (0, 1)$.

5 CBR procedure in managing computer files

In this section, we discuss the key steps related to the application of CBR to file management.

5.1 Case retrieval

Case retrieval is actually to retrieve n nearest neighboring cases from the case base, which are similar to the new problem at hand, using a predefined similarity function. There are quite a few functions available for measuring the similarity between two vectors, such as, Manhattan distance, Euclidean distance, and Mahalanobis distance (Khoshgoftaar et al. 2006). In our study, we employ the Manhattan distance, namely, the weighted summation of the similarities of all descriptive attributes as the function for measuring the similarity between two cases, namely,

$$\mathbb{S}(F_A, F_B) = \frac{1}{|\Xi|} \sum_{x \in \Xi} w_x \cdot \mathcal{S}\left(F_A^x, F_B^x\right), \tag{6}$$

where Ξ is the full set of descriptive attributes; w_x is the weight of the descriptive attribute x, which indicates its relative importance as compared to other descriptive attributes in Ξ; $\mathcal{S}\left(F_A^x, F_B^x\right)$ is the similarity between the descriptive attribute x of cases F_A and F_B.

5.2 Case reuse and revision

Given the new case F_A, after the n nearest neighboring cases have been retrieved out from the case base, they will be classified into two groups according to their final action attributes. Suppose Φ be the set of n nearest neighboring cases. Let Φ_d and Φ_p be the subsets of Φ, whose elements have `delete` and `preserve` as their final action attributes, respectively. Therefore, we have $\Phi_d \cup \Phi_p = \Phi$ and $|\Phi_d| + |\Phi_p| = n$. Next, the CBR system can recommend a final action for the new case F_A as follows:

$$F_A^{act} = \begin{cases} \texttt{delete}, & \text{if } \sum_{F_B \in \Phi_d} \frac{F_B^{crd}}{F_B^{trd}} - \sum_{F_B \in \Phi_p} \frac{F_B^{crd}}{F_B^{trd}} > \mu, \\ \texttt{preserve}, & \text{if } \sum_{F_B \in \Phi_p} \frac{F_B^{crd}}{F_B^{trd}} - \sum_{F_B \in \Phi_d} \frac{F_B^{crd}}{F_B^{trd}} > \nu, \\ \texttt{keep}, & \text{otherwise}, \end{cases} \tag{7}$$

where F_B^{crd}/F_B^{trd} is the ratio of the number of correct recommendations to that of the total recommendations of the case F_B and is thus called *correct recommendation rate*; μ and ν are positive constants indicating the preference of the file management system to deleting and preserving files, respectively. The smaller the value of μ (ν), the higher the preference of the system to delete (preserve) files.

In our CBR system, case revision mainly concerns the performance attributes of cases stored in the case base. After the final decision on a new case has been made and the corresponding action has been performed, the performance attributes of the n nearest neighboring cases in Φ that have been retrieved from the case base will be updated. Specifically, the update will be carried out as follows:

(1) For all cases contained in Φ, increase their total number of recommendations by one;
(2) If the final decision on the new case F_A is `delete`, increase the correct and incorrect recommendation counters of the cases contained in Φ_d and Φ_p by one, respectively;
(3) If the final decision on the new case F_A is `preserve`, increase the correct and incorrect recommendation counters of the cases contained in Φ_p and Φ_d by one, respectively.

5.3 Case retainment and management

Let $\hat{F}_A^{act}$ be the final action that is taken by the human user or the file management agent on the new case F_A. If $\hat{F}_A^{act}$ is different from the action F_A^{act} recommended by the CBR agent, the new case F_A and the corresponding final action $\hat{F}_A^{act}$ will be retained into the case base. Moreover, the three performance attributes of F_A will be initialized to zero and the current time of the system will be taken as its action time F_A^{fat}.

In order to enable the CBR agent to work efficiently, we need to maintain the case base in a manageable scale. For this reason, the case base is regularly examined so as to remove the cases with a high incorrect recommendation rate. Here, incorrect recommendation rate is defined as the ratio of the number of incorrect recommendations of an existing case to its total number of recommendations. Specifically, we set a threshold τ for the incorrect recommendation rate. Only those cases with an incorrect recommendation rate less than τ will be kept. All other cases will be removed from the case base.

6 Performance validation

In order to validate the developed multi-agent based file management system and, particularly, the effectiveness and efficiency of the CBR mechanism in file management, we have carried out extensive experiments. In this section, we present the experimental results. In our experiments, we first created 215 cases and stored them into the case base. These cases corresponded to two groups of deleted and preserved computer files, respectively. They were further categorized into five types, namely, document, image, audio, video, and others. Next, the file management system was employed to recommend suitable actions on another group of 287 files.

In Tables 1, 2, 3, 4, and 5, we present the experimental results corresponding to different file types, respectively. For each file type, the numbers of recommendations corresponding to the three actions, `delete`, `keep`, and `preserve`, are listed. For each action, the numbers and rates of the correct and incorrect recommendations are also provided. The bottom

Table 1 The statistic results of the recommendations made by the CBR mechanism on document files

Action	Number	Correct recommendation		Incorrect recommendation	
		Number	Rate (%)	Number	Rate (%)
`delete`	15	10	67	5	33
`preserve`	18	11	61	7	39
`keep`	21	15	71	6	29
Total	54	36	67	18	33

Table 2 The statistic results of the recommendations made by the CBR mechanism on image files

Action	Number	Correct recommendation		Incorrect recommendation	
		Number	Rate (%)	Number	Rate (%)
`delete`	9	7	78	2	22
`preserve`	13	8	62	5	38
`keep`	28	20	71	8	29
Total	50	35	70	15	30

Table 3 The statistic results of the recommendations made by the CBR mechanism on audio files

Action	Number	Correct recommendation		Incorrect recommendation	
		Number	Rate (%)	Number	Rate (%)
`delete`	27	19	70	8	30
`preserve`	20	9	45	11	55
`keep`	38	25	66	13	34
Total	85	53	62	32	38

row in each table summarizes the total number of recommendations, as well as the rates and numbers of the correct and incorrect recommendations on the corresponding file type. The rates of the correct recommendations corresponding to the three actions on different file types are also graphically presented in Fig. 5. From Tables 1, 2, 3, 4, 5 and Fig. 5 we can note that for the ordinary file types (i.e., document, image, audio, video), the agent-based file management system can provide fairly accurate recommendations, whilst for other file types the performance of the system is not good. This is due to the significant differences between the existing cases and new files under consideration, although they were all categorized as other file types.

Table 6 presents the total numbers of recommendations corresponding to the three actions on all file types. The numbers and rates of the correct and incorrect recommendations of each action on all file types are also summarized. It can be found from the last row that the total number of recommendations is 287; the number and rate of correct recommendations made by the CBR-based recommendation agent are 184 and 64%, respectively. That is to say, the developed system is fairly effective in recommending suitable actions on various types of files.

Table 4 The statistic results of the recommendations made by the CBR mechanism on video files

Action	Number	Correct recommendation		Incorrect recommendation	
		Number	Rate (%)	Number	Rate (%)
`delete`	10	6	60	4	40
`preserve`	15	11	73	4	27
`keep`	31	23	74	8	26
Total	56	40	71	16	29

Table 5 The statistic results of the recommendations made by the CBR mechanism on other file types

Action	Number	Correct recommendation		Incorrect recommendation	
		Number	Rate (%)	Number	Rate (%)
`delete`	8	3	38	5	62
`preserve`	16	7	44	9	56
`keep`	18	10	56	8	44
Total	42	20	48	22	52

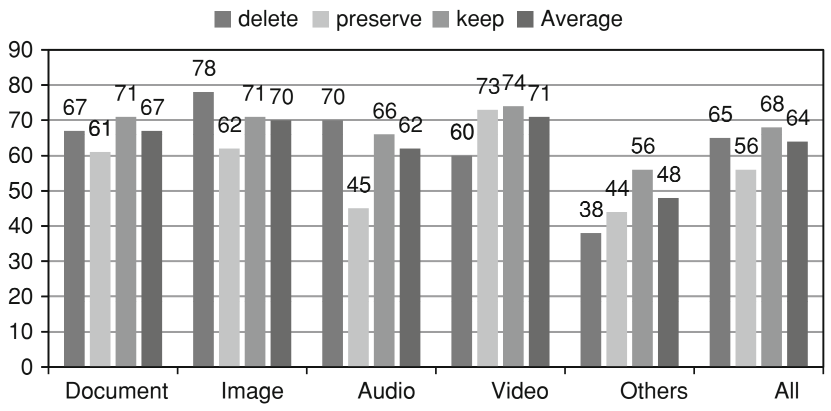

Fig. 5 The rates of the correct recommendations made by the CBR agent for different file types

Table 6 The statistic results of the recommendations made by the CBR mechanism on all file types

Action	Number	Correct recommendation		Incorrect recommendation	
		Number	Rate (%)	Number	Rate (%)
`delete`	69	45	65	24	35
`preserve`	82	46	56	36	44
`keep`	136	93	68	43	32
Total	287	184	64	103	36

7 Conclusion

As two promising artificial intelligence approaches, both IAT and CBR have been successfully applied in many different industrial and commercial domains. On the other hand, since computers have been more and more deeply involved in people's daily life, computer users are facing a progressively serious problem, namely, how to efficiently management computer files so as to not only facilitate themselves to use computer files, but also save the scare storage resource. However, although there are a lot of file management software tools available so far, none of them is able to address these crucial problems on file management: Which files should be immediately deleted after their use? Which files should be temporarily remained? Which files need to be permanently preserved? To the best of our knowledge, these issues have not yet been investigated in the open scientific literature.

To fill this gap, in this paper we explored the application of the artificial intelligence approaches to automatic file management. We presented a multi-agent based system for personal file management, where CBR is employed to recommend suitable actions on different files based on the experience knowledge learned from computer users' behaviors on managing files. Specifically, we introduced the architecture of the management system and the functionality of individual software agents. We studied the key issues of applying CBR to file management, namely, case representation, retrieval, reuse, revision, retainment, and management. Through a set of experiments and the corresponding results, we finally validated the effectiveness and efficiency of the developed file management system and the CBR mechanism on file management. We observed that the developed system can well recommend suitable actions on various computer files. In the future, we will examine the potential correlation among different descriptive attributes of cases so as to make the case representation more efficient. We will also explore other potential functions for measuring the similarity between different cases.

Acknowledgement This work is supported by the Seventh Framework Programme of the European Union under grant (FP7-ICT-216746).

References

Braubach L, Lamersdorf W, Pokahr A (2003) Jadex: implementing a BDI-infrastructure for JADE agents. EXP—In Search of Innovation 3(3):76–85
Burke EK, MacCarthy BL, Petrovic S, Qu R (2006) Multiple-retrieval case-based reasoning for course timetabling problems. J Oper Res Soc 57:148–162
Carlton GH (2005) A critical evaluation of the treatment of deleted files in microsoft windows operation systems. In: Proceedings of the 38th Hawaii international conference on system sciences (HICSS'05), pp 1–8
Chmiel K, Gawinecki M, Kaczmarek P, Szymczak M, Paprzycki M (2005) Efficiency of JADE agent platform. Sci Progr 13(2):159–172
Craw S, Wiratunga N, Rowe RC (2006) Learning adaptation knowledge to improve case-based reasoning. Artif Intell 170(16–17):1175–1192
Delany SJ, Bridge D (2006) Textual case-based reasoning for spam filtering: a comparison of feature-based and feature-free approaches. Artif Intell Rev 26(1–2):75–87
Delany SJ, Cunningham P, Coyle L (2005) An assessment of case-based reasoning for spam filtering. Artif Intell Rev 24(3–4):359–378
Eder J, Krumpholz A, Biliris A, Panagos E (2000) Self-maintained folder hierarchies as document repositories. In: Proceedings of the international conference on digital libraries: research and practice (ICDL'00), pp 400–407
Fan X, Liu Q, Ng PA (1997) A multimedia document filing system. In: Proceedings of the IEEE international conference on multimedia computing and systems (ICMCS'97), pp 492–499

Glasgow J, Kuo T, Davies J (2006) Protein structure from contact maps: a case-based reasoning approach. Inf Sys Front 8(1):29–36

Hellingrath B, Bohle C, van Hueth J (2009) A framework for the development of multi-agent systems in supply chain management. In: Proceedings of the 42nd Hawaii international conference on system sciences, pp 1–9

Iglesias R, Ares F, Ferneindez-Delgado M, Rodriguei JA, Bregains J, Barrol S (2008) Element failure detection in linear antenna arrays using case-based reasoning. IEEE Antennas Propag Mag 50(4):198–204

Jennings NR (2000) On agent-based software engineering. Artif Intell 117(2):277–296

Jennings NR (2001) An agent-based approach for building complex software systems. Commun ACM 44(4):35–41

Jennings NR, Sycara K, Wooldridge M (1998) A roadmap of agent research and development. Auton Agents Multi-Agent Sys 1(1):7–38

Joukov N, Zadok E (2005) Adding secure deletion to your favorite file system. In: Proceedings of the 3rd IEEE international security in storage workshop (SISW'05)

Khanum A, Shafiq MZ (2006) Facial expression recognition system using case based reasoning. In: Proceedings of the 2006 international conference on advances in space technologies, pp 147–151

Khoshgoftaar TM, Seliya N, Sundaresh N (2006) An empirical study of predicting software faults with case-based reasoning. Softw Qual J 14(2):85–111

Lind J (2000) Issues in agent-oriented software engineering. In: Proceedings of the first international workshop on agent-oriented software engineering (AOSE 2000), pp 45–58

Luck M, McBurney P, Shehory O, Willmott S (2005) Agent technology: computing as interaction (A Roadmap for Agent Based Computing). AgentLink

Mahalingam M, Tang C, Xu Z (2003) Towards a semantic, deep archival file system. In: Proceedings of the 9th IEEE workshop on future trends of distributed computing systems (FTDCS'03), pp 115–121

Mutka MW, Ni LM (1992) Managing personal files across independent file management units. In: Proceedings of the 3rd workshop on future trends of distributed computing systems, pp 254–261

Perner P (2002) Are case-based reasoning and dissimilarity-based classification two sides of the same coin?. Eng Appl Artif Intell 15(2):193–203

Pous C, Caballero D, Lopez B (2008) Diagnosing patients combining principal components analysis and case based reasoning. In: Proceedings of the 8th international conference on hybrid intelligent systems (HIS'08), pp 819–824

Ross S, Fang L, Hipel KW (2002) A case-based reasoning system for conflict resolution: design and implementation. Eng Appl Artif Intell 15(3–4):369–383

Sun H, Wang W, Li Y, Huang T (2008) Dynamic business alliance and its construction based on multi-agent system. In: Proceedings of the 4th international conference on wireless communications, networking and mobile computing (WiCOM'08), pp 1–5

Watson I, Marir F (1994) Case-based reasoning: a review. Knowl Eng Rev 9(4):355–381

Wedde HF, Korel B, Brown WG, Chen S (1990) Distributed management of replicated and partitioned files under DRAGON SLAYER. In: Proceedings of the 14th annual international computer software and applications conference (COMPSAC'90), pp 436–441

Yang S, Gechter F, Koukam A (2008) Application of reactive multi-agent system to vehicle collision avoidance. In: Proceedings of the 20th IEEE international conference on tools with artificial intelligence (ICTAI'08), pp 197–204

You LL, Pollack KT, Long DDE (2005) Deep store: an archival storage system architecture. In: Proceedings of the 21st international conference on data engineering (ICDE'05), pp 804–815

Zambonelli F, Omicini A (2004) Challenges and research directions in agent-oriented software engineering. Auton Agents Multi-Agent Sys 9(3):253–283