Towards Task-Organised Desktop Collections

Yukun Li
School of Information
Renmin University of China
liyukun@ruc.edu.cn

David Elsweiler
Department of Computer Science,University of Erlangen
david@elsweiler.co.uk

Xiaofeng Meng
School of Information
Renmin University of China
xfmeng@ruc.edu.cn

ABSTRACT

In this paper we promote the idea of automatic task-based document organisation. To make this possible we present a simplified task model and evaluate a number of algorithms for detecting which documents are associated with particular tasks. Our findings demonstrate the feasibility of such an approach, but work must be done to improve the performance for practical implementation.

1. INTRODUCTION

As people acquire ever more information as a result of personal and work activities, the management of this information becomes a serious problem and an important research issue [6]. Previous literature suggests that the tasks people perform and the activities associated with personal information plays an important role in how the information will be managed and re-found [4, 11]. There are also a number of anecdotal scenarios that highlight the importance of user task and activities in Personal Information Management (PIM) behaviour:

(1) We know that people often multi-task and experience difficulties when switching between tasks [3]. To support multi-tasking it would be useful for the user to have access to resources associated with each task; (2) When restarting a personal computer, especially after a change in workplace (i.e. from office to home) a user may need to access the files related to specific tasks in order to continue his work; (3) When starting a new task similar to a task already completed, it may be helpful to access documents associated with the previous task for reference purposes; (4) When an experienced user wants to help someone with less experience complete a task, it is often useful, for demonstration purposes, to re-find personal documents associated with the completion of this or a similar activity in the past. (5) When writing a progress report or summary of work completed, it is often useful to review completed activities retrospectively and see which documents have been created, used or modified.

To support these kinds of scenarios PIM systems need to be able to associate documents with user activities. Currently the only way to achieve this is to rely on user annotation and filing. Nevertheless, there is a large body of evidence suggesting that people are not willing or able to achieve a consistent organization, which meets all of their needs over long-periods of time [8, 9, 13].

It would be very advantageous if tasks were able to be modelled in such a way that they could be automatically detected and appropriate documents and data items could be associated with activities. It is possible, for example, that such a model could be used to implement a task-based replacement for or enhancement to the the traditional desktop metaphor. Nevertheless, there several challenges that need to be faced before such as model can be realised.

Firstly, it is difficult to define the concept of a task. Tasks can be considered at various granularities e.g. a project could be considered a task at a high-level, but would naturally consist of many sub-tasks and sub-sub tasks. Tasks can also be of various complexities [2]. Evaluating any model created or algorithm used to detect tasks is also challenging because in addition to the many problems associated with PIM evaluations, such as personalisation and privacy problems [4], there are no publicly available data sets, nor any available benchmarks or frameworks for evaluation.

In this paper we present our work in addressing these challenges. We formally define the concept of a user task and use the definition as the foundation for a task-based model for PIM. We also outline our thoughts on evaluation and describe some early results from experiments performed to test the performance of various algorithms which associate resources with tasks.

1.1 Related Work

PIM is the area of research concerned with how people store, manage and re-find information [6]. It is a multi-disciplinary field and researchers have been actively trying understand user behaviour, such as how people interact with information and tools [9, 13], what psychological factors are important [1] and how improved tools can support user behaviour and needs [10, 5].

A large amount of PIM behaviour is performed on desktop computers, where the standard PIM model is based on the office metaphor of files and folders. Several scholars have identified the limitations of this model and suggested moving to other means of interacting with information [7, 5]. One suggested method has been to organise information based on the activities or tasks the user performs. Studies have shown the importance of activities and tasks to PIM, including behavioural strategies to allow tasks to be managed [13] and indicating that while working, people regularly need to switch between concurrent tasks [3] and have difficulty managing resources as a result. The general consensus is that the desktop metaphor provides inadequate resources for task management and switching [11] and as a result,

several prototype systems have been designed to assist with these situations, either by visualising resources in different ways e.g.[11] or making tasks the basis for organisation [12].

The common theme and, in our opinion, the major limitation of the task-based solutions proposed to date is that they all require the user to indicate which resources are associated with each task, which places the cognitive burden on the user in the same way the desktop metaphor does [8]. In this paper we explore methods to automatically associate resources with tasks. We first present a definition of a task and a model from which resources can be associated with tasks and continue propose and evaluate a number of algorithms for automatic association.

2. TASK MODEL

The basic building block for our tasks is a personal data item (PDI), which we define as *an item which has been created, accessed or modified by a person and is the result of user interaction.*

According to this definition, a file, an email or a folder can all be regarded as PDIs. What the definition also makes explicit is that PDIs only refer to items that the user has interacted with and that have not automatically been generated by software. This is important as in this work we focus on illustrating a task model and methods for identifying tasks based on desktop collections. The aim of any model would be to exploit user recollections for activities associated with PDIs and users will not remember any item with which they have had no explicit interaction.

A task has three basic elements: a goal (i.e. the thing to be accomplished), process (i.e. the activities performed to achieve the goal), and time (the period of time taken to complete the task). Each of these elements must be included and formalised in the task definition.

User activities with desktop computers can be divided into two types based on their interactions. ***Read-only activities*** involve only accessing a PDI, and include activities, such as reading documents, listening to music, etc.. The second type of activity, ***creative activities***, involve generating new or updating existing PDIs. Thus, the goal of a creative activity can be materialized as the set of PDIs modified by the user. In this paper, our focus is on this second category of tasks. According the above criteria, a personal task can be described as follows:

Definition 2.1 (Personal Task). . *A personal task PT is described as a 4-tuple (TD, DI, OL, TL), where TD represents a written description of the task and is described as a vector of tokens. DI represents the related data items, and is denoted as a 2-tuple (GI, RI), where GI (goal items) is a set of items generated by users during the process of completing the task, and RI (referenced items) is a set of items accessed in order to complete the task. OL (operations list) is a sequential list user operations performed to complete the task. TL describes the life-cycle of the task and is denoted as a 2-tuple (TS, TE), where T_s is the start time of the task and T_e is the end time.*

Naturally, as mentioned above, tasks can be of varying levels of complexity. For example, notifying colleagues about an upcoming meeting would be a task requiring the creation of only one item, i.e. an email. Preparing a grant application, on the other hand, could also be considered a task, but may involve the creation or modification of multiple items. Consequently we consider two types of tasks: *Simple Tasks*, which have a single goal item and *Complex Tasks*, which have multiple goal items.

In practical terms, a complex task can be regarded as the combination of many simple tasks. Therefore, if simple tasks can be identified then this could form the basis task identification in general. For this reason we focus here on identifying simple tasks.

3. IDENTIFYING PERSONAL TASKS

According to the definition above, there are several elements of a simple task that need to be identified: the task description, the life-cycle of the task, referenced PDIs and a task goal PDI. From this list, detecting referenced PDIs is the most challenging problem. For simple tasks, any created or modified PDI could be considered a task goal item. A task description can be generated by applying techniques, such as TF-IDF, to the PDI content or utilising a file or folder name (for non-text-based PDIs). The life-cycle information can also be easily attained by taking the created and modified times of the goal item (GI).

Below, we will outline a number of basic methods for identifying reference files based on PDI properties and patterns of user interactions. We evaluate the performance of the various methods in Section 4.

3.1 Life-cycle based method

A simple method of detecting PDIs associated with tasks is to take all files accessed within the life-cycle of a task as its references. This method will achieve perfect recall i.e. all of the files associated with a task will be detected, but the fact that people multi-task and continue tasks over long time periods will inevitably result in very low precision i.e. many inappropriate files being taken as task references. Nevertheless, the high recall property makes the approach a useful baseline algorithm for evaluations.

3.2 Directory-based method

We know from studies of folder organisations that people often organise their information items based on activity [9, 13]. Thus, a simple approach to associating PDIs with tasks is to utilise the information implicitly provided by the user through his folder structure. Given a task goal file, we can take all files located in the same folder (as well as the folder itself) as its references. An obvious limitation of the method is that files can be organised in other ways (e.g. time, people etc.) so it is likely that in some cases appropriate RIs will be located across folders – such references will not be detected using this method.

3.3 Sequential distance-based method

Another assertion we can make regarding user behaviour is that items accessed and modified within similar time periods relate to the same task. If this is true then the sequential distance – the number of items accessed or modified between two items in a sequential access list – will reflect, to some degree, the relationship between the items.

This method is simple to implement, but also has limitations. We expect it to work well when the user does not multi-task, but poorly for multi-tasking situations. Finding an appropriate threshold distance is also a problem. We assume that a small threshold will lead to higher precision

but lower recall and a larger threshold will lead to a low precision but higher recall. We provide some data regarding threshold selection in the experimental section below.

3.4 Operational pattern-based method

A further assertion we can make about user-behaviour is that after referring to a file, the user will modify the goal item of the task. If this is the case we can expect modify operations on the goal item to be surrounded by read operations of reference items for that particular goal item. Inspection of user interaction logs (see experiment below), seems evidence this assertion. We can exploit this with the following algorithm

Algorithm 1 Operational Pattern Based Algorithm

Input: An existing access list $L' = X_1, X_2, ..., X_n$; An existing task set TS; Latest accessed file X_{n+1};
Output: An updated task set TS.

procedure $Identify\ Simple\ Task(L', TS, X_{n+1})$
  if then $X_{n+1}.operation = "Modify"$
    if then $\nexists t \in TS \wedge t.goalitem = X_{n+1}.item$
      Create a new task t
      Add $X_n.item$ into t.reference
    else
      Find a task t where $t.goalitem = X_{n+1}.item$
      Add $X_n.item$ into t.reference
      Find $MSL\ L'' = (X_k, X_{k+1}, ..., X_{n+1})$
      Add $X_i.item (k+1 \leq i \leq n)$ into t.references
    end if
  end if
end procedure

When a modification operation is detected,i.e. a GI is processed, the references for that task will be updated. First, the algorithm will find the latest read operation for the GI within the access list L'. Following this it checks to see if two records in L' exist, which point to the same DI. If not, the access list is denoted as a Minimum Sequential Loop(MSL) and all DIs accessed within this MSL are regarded as references of the GI. Unlike the sequential distance-based method, here no threshold value needs to be specified in advance. However, like the SD method this approach has the disadvantage that multi-tasking behaviour may negatively affect performance.

4. EVALUATING PERFORMANCE

A dataset collected via a naturalistic investigation formed the basis of our evaluation. By developing and deploying a custom-designed piece of software built based on APIs for Microsoft Windows operating system, we captured user file accesses during the course of normal PC usage. We recorded two types of operation: "read-operations" where items were accessed or read and "modify-operations", where items were newly created or modified. For each operation we stored the associated file name, the directory-path and a timestamp.

8 participants (4 male, 4 female, aged between 25 and 40), volunteered to take part over a period of approximately one year. The participants were all researchers or research students at a major Chinese university and although they represent a relatively homogeneous population, they are all busy people who struggle with multi-tasking and PIM in general. The data represented their activities with their office computers.

In total 54,545 operations were recorded (avg per participant =6818 st dev =3947). 79% of the operations were reads and and 21% were modify operations.

To create a "gold-standard" from which to compare the performance of the algorithms, we conducted a second experimental phase where the same participants were asked to manually indicate the referenced items for a given set of goal items. We selected 10 goal items per participant so that those chosen included both files modified often and only a few times. To account for the difficulties in retrospectively annotating referenced files, the participants used software, which showed a goal item and a list of potential referenced items (selected by the life-cycle based method). The participants had access to meta-data about each of the items and could also open the files to examine the content before deciding if it was a referenced file. Additionally, we asked the participants to explain the relationship between the item and the goal item. We used these data as a means to evaluate the algorithms described in Section 3.

5. RESULTS AND DISCUSSION

Figure 1 depicts the performance achieved by the various algorithms. As these graphs show, it was possible to attain high recall scores – all of the algorithms achieved average scores of 0.71 and above. However, precision was harder to attain with the best performance being achieved by the OP method (0.43).

The Directory-based approach achieved the lowest recall (0.71), with some referenced files evidently being stored in different directories. Figure 1(b) shows that some participants tended to place task-related items in the same folder (e.g. user 5) and this allowed high recall to be achieved. What is also clear from the low precision scores, is that all of the participants had files in the examined directories, which they did not consider to be related by task.

Figure 1(f) shows how the sequential distance threshold (sd) influenced the performance for the SD method. Confirming our hypotheses, when the threshold=1, we achieved the optimum F-score (0.34) and precision (0.27), but as the threshold is increased, the F-score and precision deteriorated as recall improved. This result indicates that in a practical implementation of the algorithm, it would not be necessary to use a high threshold to achieve best performance.

Figure 1(a-d) shows how the algorithms performed across users, demonstrating that it was easier to achieve higher precision with some users e.g. user 6. We hypothesize that users, for which better performance could be attained, tend to multi-task less often. If this is true it highlights a paradoxical situation where the people most likely to need support are also the most difficult to provide support for. This is an obvious weakness in the methods proposed.

What our results do show, however, is that there is potential for algorithms to automatically related items by task. Our relatively simple algorithms were able to achieve good recall although improvement is needed in terms of precision. To achieve this improvement we need to investigate ways of combining evidence from the various sources exploited in the basic algorithms presented here. Even with current performance levels, we believe that, if embedded within an appropriate user interface, the algorithms could be useful in assisting users to manually organise their documents. These

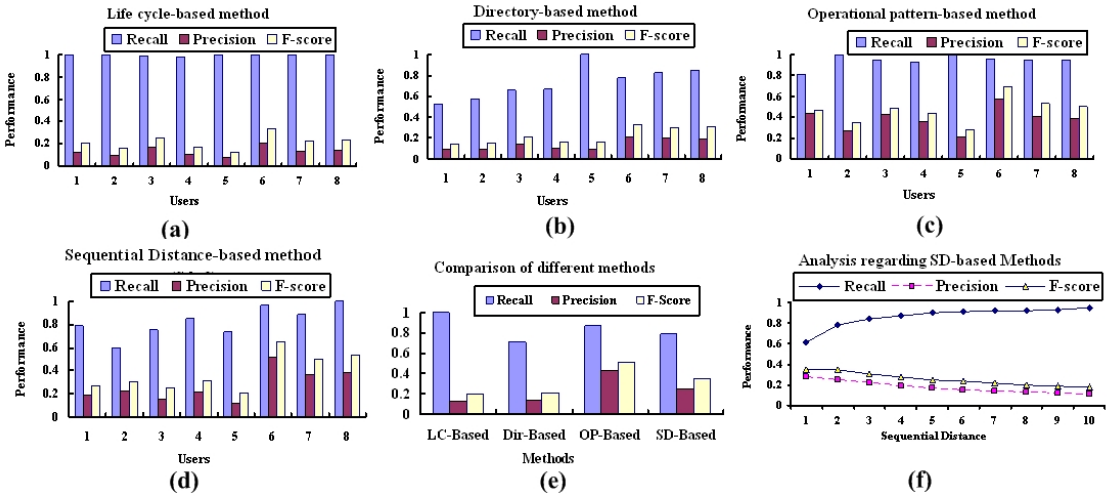

Figure 1: The Performance Achieved by the Various Algorithms

two threads represent our future plans for this work.

6. CONCLUSIONS

In this paper we have used examples from the literature to demonstrate the potential benefits of a task-oriented approach to PIM. We suggested that to realise these benefits tasks should be detected automatically and to achieve this we proposed a task model and several methods of detecting resources associated with tasks. Our model abstracts and simplifies the concept of a task, allowing the performance of the proposed algorithms to be evaluated. The evaluation results suggest that it may be feasible to achieve an automatic means of task-detection, but work must be done to improve the performance for practical implementation.

7. ACKNOWLEDGMENTS

This research was partially supported by the grants from the Natural Science Foundation of China (No.60833005); the National High-Tech Research and Development Plan of China (No.2009AA011904); and the Doctoral Fund of Ministry of Education of China (No. 200800020002).

8. REFERENCES

[1] D. Barreau, *Special issue on the social and psychological aspects of personal information management*, Journal of Digital Information **10** (2009), no. 5.

[2] K. Byström and K. Järvelin, *Task complexity affects information seeking and use*, Information Processing and Management **31** (1995), no. 2, 191–213.

[3] M. Czerwinski, E. Horvitz, and S. Wilhite, *A diary study of task switching and interruptions*, CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems (New York, NY, USA), ACM Press, 2004, pp. 175–182.

[4] D. Elsweiler and I. Ruthven, *Towards task-based personal information management evaluations*, SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM Press, 2007, pp. 23–30.

[5] E. Freeman and D. Gelernter, *Lifestreams: a storage model for personal data*, SIGMOD Record (ACM Special Interest Group on Management of Data) **25** (1996), no. 1, 80–86.

[6] W. Jones and J. Teevan (eds.), *Personal information management*, Seattle: University of Washington Press, 2007.

[7] V. Kaptelinin and M. Czerwinski, *Beyond the desktop metaphor designing integrated digital work environments*, MIT Press, 2007.

[8] M.W. Lansdale, *The psychology of personal information management.*, Appl Ergon **19** (1988), no. 1, 55–66.

[9] T. W. Malone, *How do people organize their desks?: Implications for the design of office information systems*, ACM Trans. Inf. Syst. **1** (1983), no. 1, 99–112.

[10] G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzich, *Data mountain: using spatial memory for document management*, UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology (New York, NY, USA), ACM Press, 1998, pp. 153–162.

[11] G. Robertson, G. Smith, B. Meyers, P. Baudisch, M. Czerwinski, E. Horvitz, D. C. Robbins, and D. Tan, *Beyond the desktop metaphor*, ch. Explorations in Task Management on the Desktop, pp. 101–138, MIT-Press, 2006.

[12] S. Stumpf and J. Herlocker, *Tasktracer: Enhancing personal information management through machine learning*, Proc. Workshop on Personal Information Management, SIGIR, 2006.

[13] S. Whittaker and C. Sidner, *Email overload: exploring personal information management of email*, CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems (New York, NY, USA) (M. J. Tauber, ed.), ACM Press, 1996, pp. 276–283.