Fewer Clicks and Less Frustration: Reducing the Cost of Reaching the Right Folder

Xinlong Bao
School of EECS
Oregon State University
Corvallis, OR 97331
+1 541 737-1646
bao@eecs.oregonstate.edu

Jonathan Herlocker
School of EECS
Oregon State University
Corvallis, OR 97331
+1 541 737-8894
herlock@eecs.oregonstate.edu

Thomas G. Dietterich
School of EECS
Oregon State University
Corvallis, OR 97331
+1 541 737-5559
tgd@eecs.oregonstate.edu

ABSTRACT
Helping computer users rapidly locate files in their folder hierarchies has become an important research topic in today's intelligent user interface design. This paper reports on FolderPredictor, a software system that can reduce the cost of locating files in hierarchical folders. FolderPredictor applies a cost-sensitive prediction algorithm to the user's previous file access information to predict the next folder that will be accessed. Experimental results show that, on average, FolderPredictor reduces the cost of locating a file by 50%. Another advantage of FolderPredictor is that it does not require users to adapt to a new interface, but rather meshes with the existing interface for opening files on the Windows platform.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning – *Knowledge acquisition*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *User-centered design*.

General Terms
Design, Human Factors.

Keywords
Tasks, activities, prediction, recommendation, user interface, folders, directories, shortcuts, machine learning, intelligent user interfaces.

1. INTRODUCTION
Computer users organize their electronic files into folder hierarchies in their file systems. But unfortunately, with the ever-increasing numbers of files, folder hierarchies on today's computers have become large and complex [2]. With large numbers of files and potentially complex folder hierarchies, locating the desired file is becoming an increasingly time-consuming operation. Some previous investigations have shown that computer users spend substantial time and effort in just finding files [1, 10, 11]. Thus, designing intelligent user interfaces that can help users quickly locate the desired files has emerged as an important research topic.

Previous research on helping users find files has focused on building Personal Information Management (PIM) systems in which documents are organized by their properties [5, 7]. These properties include both system properties, like name, path and content of the document, and user-defined properties that reflect the user's view of the document. In these systems, users can search for files by their properties using search engines. Although these search engines can be effective in helping the user locate files, previous user studies have indicated that instead of using keyword search, most users still like to navigate in the folder hierarchy with small, local steps using their contextual knowledge as a guide, even when they know exactly what they were looking for in advance [1, 9, 10, 20].

In this paper, we try to address the file-locating problem from another perspective, using a system that we call the *FolderPredictor*. The main idea of FolderPredictor is that if we have observations of a user's previous file access behavior, we can recommend one or more folders directly to the user at the moment he/she needs to locate a file. These recommended folders are predictions – the result of running simple machine learning algorithms on the user's previously observed interactions with files.

Ideally we want to identify when the user has just started to look for a file and provide a shortcut to likely choices for files. In today's graphical user interfaces, there are several user actions that strongly indicate the user is or will be initiating a search for a file. These include the display of a file open/save dialog box or the opening of a file/folder browsing application such as Windows Explorer. Our approach intervenes in the first case – the file open/save dialog box.

FolderPredictor presents predicted folders by changing the default folder of the open/save file dialog that is displayed to computer users from within an application. It also provides shortcuts to secondary recommendations, in case the top recommendation is not correct. An important advantage of FolderPredictor is that it

IUI'06, January 29–February 1, 2006, Sydney, Australia.

reduces user cost without requiring users to adapt to a new interface. Users have nothing new to learn.

The paper is organized as follows. In the next section, we introduce the TaskTracer hypothesis. Section 3 describes the bases of our folder predictions. Section 4 briefly introduces how FolderPredictor acquires data from TaskTracer. Section 5 presents the cost-sensitive prediction algorithm used by FolderPredictor to make predictions. Section 6 introduces the user interface of FolderPredictor and the instrumentation for presenting predictions in the open/save dialog box. Section 7 reports the experimental results, which establish that FolderPredictor reduces the user's cost for locating files. Section 8 discusses some related work in email classification. Finally, we conclude the paper with some discussion in section 9.

2. THE TASKTRACER HYPOTHESIS

FolderPredictor monitors user activity to make predictions to optimize the user experience. Previous research in intelligent "agents" has explored this kind of approach [8, 12, 15]. However, one of the challenges faced by these intelligent agents is that users are highly multitasking, and they are likely to need access to different files depending on the exact task they are performing. For example, a professor may have a meeting with a student for one hour and then switch to writing a grant proposal. The notes file for the student meeting and the files for the grant proposal are normally stored in different folders. After the professor switches to working on the grant proposal, a naïve agent might assume that the student notes are still the most likely recommendation, because they were the most recently accessed. This recommendation would fail, because the agent is not taking into consideration the context of the user's current task.

Previous work on intelligent agents or assistants has attempted to produce more context-aware predictions by analyzing the content of the documents that the user is currently accessing to generate an estimated keyword profile of the user's "task" [3]. This profile is then employed as a query to locate files with similar keyword profiles. This approach is limited because a) it cannot recommend resources that do not have text associated with them, b) there are substantial ambiguities in text, resulting in significant uncertainty in mapping from text profiles to user tasks, and c) the active window may not contain sufficient text. In the last case, the agent would be forced to scan recently accessed documents to generate text profiles – documents that may have been accessed as part of a previous task.

FolderPredictor is built upon our TaskTracer system [6, 19] – a research software system designed to help multi-tasking computer users with interruption recovery and knowledge reuse. There are three core hypotheses behind TaskTracer:

1) All information workers break their work into discrete units to which they can give names – we call these *tasks*.

2) At any moment in time, the user is working on only one task.

3) Knowledge of the user's current task will substantially reduce the uncertainty in predicting what a user is trying to do.

Users define new tasks in the TaskTracer system via an application called the TaskExplorer. Users can then indicate their current task through two mechanisms, which are shown in Figure 1. Users can switch to the TaskExplorer and select the task from their defined hierarchy of tasks, or they can set the task without leaving the current window by clicking on a UI component that appears in the title bar of every window. The component in the title bar is called the TaskTitleBar, and it displays the task most-recently specified by the user. The user can switch tasks by manually selecting another task from the drop-down menu of TaskTitleBar or by typing in TaskTitleBar box (with auto-completion).

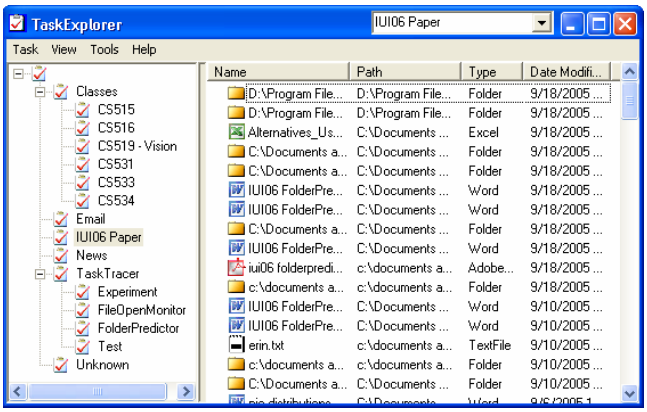

Figure 1. TaskExplorer and TaskTitleBar.

We also have a system called TaskPredictor, which can automatically detect task switches by observing the activity of the user [19]. If a probable task switch is detected, the user can be notified, actively or peripherally. Alternatively the task can be automatically changed if task prediction confidence is high enough.

TaskTracer collects detailed records of users' activities and resources accessed, and associates each activity event with the current declared task. Thus, by receiving user activities tagged with task information from TaskTracer, FolderPredictor can make task-aware predictions of what folder a user is likely to want to access based on the user's previous accesses for this task.

3. APPROACH

Our approach assumes that computer users, for the most part, separate files for different tasks into different folders. We further assume that, for the most part, the working set of folders for a task is relatively small. Given such assumptions, the paths of the files that the user has accessed when they were working on a task are useful resources for making folder predictions for future accesses for that task. For example, imagine that, during one day, a student opens and saves files under the folders "C:\Classes\CS534\Homework" and "C:\Classes\CS534\Presentation" when he is working on a task named "CS534". The next day, when he returns to working on the CS534 task, it should be useful to recommend these two folders or even the parent folder "C:\Classes\CS534".

FolderPredictor generates its predictions by applying a simple machine learning method to a stream of observed file open/save events. Each of these events includes the name of the folder

containing the file that was opened or saved. For each task, FolderPredictor maintains statistics for each folder – how many times the user opened files from it or saved files to it.

A statistical approach to making folder predictions is important for two reasons: 1) more frequently accessed folders should be more probable predictions and 2) users sometimes access the wrong files, or forget to specify that they have changed task. In the second case, the misleading events will add to the statistics. If observed accesses to a particular folder are really just noise, then we are unlikely to observe repeated future accesses to that folder over time. Thus these folders should have relatively low access frequencies. We can use this information to filter out these folders from recommendations.

Another factor that should be considered is recency. Our hypothesis is that a user is more likely to need to access recently-accessed folders. For example, a programmer working on a big project may need to access many different folders of source code, working on them one by one, but accessing each folder multiple times before moving on to the next folder. Thus, the folders with older access times should be quickly excluded from the predictions when the programmer begins to work on source code under new folders. To achieve this, we have incorporated a recency weighting mechanism into FolderPredictor. Instead of keeping a simple count of how many times a folder is accessed, we assign a recency weight w_i to each folder f_i. All weights are initially zero. When a file in folder f_i is opened or saved while working on a task, the weights of all the folders that have been accessed on that task are multiplied by a real number α between 0 and 1, and w_i is then increased by 1. α is called the *discount factor*. Multiplying by the discount factor exponentially decays the weights of folders that have not been accessed recently. When $\alpha = 0$, only the most recently-accessed folder will have a nonzero weight, and it will always be predicted. When $\alpha = 1$, no recency information is considered, and weights are not decayed. Experiments show that a discount factor in the range [0.7, 0.9] performs the best. Another benefit of recency weighting is that folders erroneously identified as relevant due to noisy data are excluded from consideration quickly, because their weights decrease exponentially.

In the implementation of FolderPredictor, we apply an incremental update method to maintain the folder weights. The first time FolderPredictor is run on a computer, historical TaskTracer data are used to build the initial list of folders and associated weights for each task. This information is stored in FolderPredictor's private database. Then FolderPredictor incrementally updates this database as new open or save events arrive, until the FolderPredictor is shut down. The next time FolderPredictor is started, it updates its database using only the events that have been added since the last time FolderPredictor was shut down. This incremental update method helps FolderPredictor keep its data up-to-date without any perceivable delay in prediction time or startup (except the first time it is run).

4. DATA COLLECTION

The TaskTracer system employs an extensive data-collection framework to obtain detailed observations of user activities. It instruments a wide range of applications under the Windows XP operating system, including Microsoft Office, Internet Explorer, and Adobe Acrobat. *Listener* components are plug-ins into the applications that capture user interaction events and send them to the *Publisher* as *Events*, which are XML strings. One event of interest is the TaskBegin event, which is sent to the Publisher when the user switches tasks. The Publisher stores these events in its database as history data and also distributes them to *Subscriber* applications that need to react to events online. FolderPredictor is one of the subscriber applications of TaskTracer. This *Publisher-Subscriber Architecture* is shown in Figure 2.

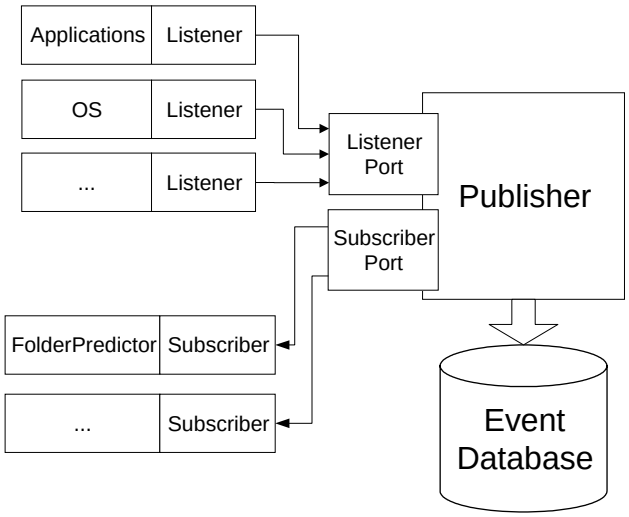

Figure 2. Publisher-Subscriber Architecture.

TaskTracer provides a variety of events to its subscriber applications, some of which are useful for making folder predictions.

5. COST-SENSITIVE PREDICTION ALGORITHM

The main goal of FolderPredictor is to reduce the cost of locating files. To achieve this, FolderPredictor applies a cost-sensitive algorithm to make predictions.

In this paper, we assume that the user navigates in the open/save file dialog using a mouse. The number of "clicks" necessary to reach the destination folder is used as our measure of cost to the user. One "click" can lead the user from the currently selected folder to its parent folder or to one of its sub-folders.

We could just recommend the folder having the highest weight. However, while that might maximize the chance that we pick the best possible folder, it may not minimize the average cost in clicks. To do this, we may want to sometimes select an ancestor folder. We will motivate the need for this decision by an example. Suppose a student has a folder hierarchy as shown in the tree structure in Figure 3. His files are stored in the bottom level folders, such as "Part3" and "Hw2". When he worked on the "CS534" task, he accessed almost all of the bottom level folders with approximately similar counts. In this circumstance, predicting a bottom level folder will have a high probability of being wrong because all the bottom level folders are equally probable. This will cost the student several more "clicks" to reach his destination folder – first to go up to an ancestor folder and then go back down to the correct child. On the other hand,

predicting a higher level folder, such as "Presentations" or "Homeworks" or even "CS534", may not be a perfect hit, but it may reduce the cost in half – the student only needs to go downward to reach his destination folder. Furthermore, we believe that incorrectly predicting a leaf node will be on average more frustrating for users than picking an ancestor node that is an incomplete path to the desired folder.

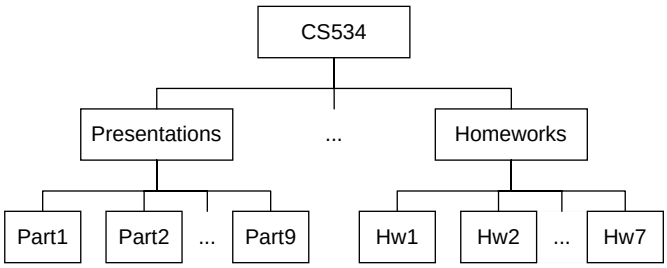

Figure 3. An example folder hierarchy. Each node of this tree denotes a folder. Child nodes are sub-folders of their parent node.

Based on this idea, we developed the following cost-sensitive prediction algorithm.

Algorithm 1 (Cost-sensitive Prediction Algorithm)

Input: A finite set $F = \{f_1, f_2, \ldots, f_m\}$, where f_i is a folder with a positive weight w_i, $1 \leqslant i \leqslant m$.

Output: Three recommended folders (descending in predicted preference).

Step 1. Compute a probability p_i for each f_i by normalizing the weights: $p_i = w_i \Big/ \sum_{i=1}^{m} w_i$.

Step 2. Build the hypothesis set: $H = \bigcup_{i=1}^{m} H_i$, where H_i is the set of all ancestors of f_i, including f_i itself.

Step 3. Compute a distance $L(h, f)$ for each $h \in H$ and $f \in F$: $L(h, f)$ = the length of the path from h to f in the tree-structured folder hierarchy.

Step 4. Return the three different folders a_1, a_2, a_3 from H that minimize the expected cost:

$$\sum_{i=1}^{m} p_i \bullet \mathbf{min}\{L(a_1, f_i),\ L(a_2, f_i)+1.0,\ L(a_3, f_i)+1.0\}$$

In Step 4 of the above algorithm, the cost of getting to a folder f_i from a prediction (a_1, a_2, a_3) is computed as $\mathbf{min}\{L(a_1, f_i),\ L(a_2, f_i)+1.0,\ L(a_3, f_i)+1.0\}$, because of the following facts:

1) a_1 will be set as the default folder of the open/save file dialog. This means that the user will be in folder a_1 with no extra "click" required. Therefore, $L(a_1, f_i)$ is the cost if the user navigates to f_i from a_1.

2) a_2 and a_3 will be shown as shortcuts to the corresponding folders in the "places bar" on the left side of the open/save file dialog box. The user must execute one "click" on the places bar if he wants to navigate to f_i from a_2 or a_3. Therefore, an extra cost 1.0 is added.

3) We assume the user knows which folder to go to and how to get there by the smallest number of clicks. Therefore, the cost of a prediction (a_1, a_2, a_3) is the minimum of $L(a_1, f_i)$, $L(a_2, f_i)+1.0$, and $L(a_3, f_i)+1.0$.

FolderPredictor runs this algorithm whenever the user switches task and whenever the folder weights are updated. The returned three predicted folders are stored for recall by the FolderPredictor UI when needed.

6. FolderPredictor UI

A basic design principle of FolderPredictor is simplicity. This basic principle greatly influences the UI design – FolderPredictor actually has *no independent UI*.

FolderPredictor shows its predictions directly in the open/save file dialog provided by the Windows operating system. It hooks into the native Windows environment and is transparent to the user. Therefore, FolderPredictor carries no overhead for the user to learn additional interactions.

Figure 4 shows an example open file dialog box with FolderPredictor running. The three predicted folders are shown as the top three icons in the "places bar" on the left. The user can jump to any of them by clicking on the corresponding icon. The most probable folder is also shown as the default folder of the dialog box so that the display will show this folder initially.

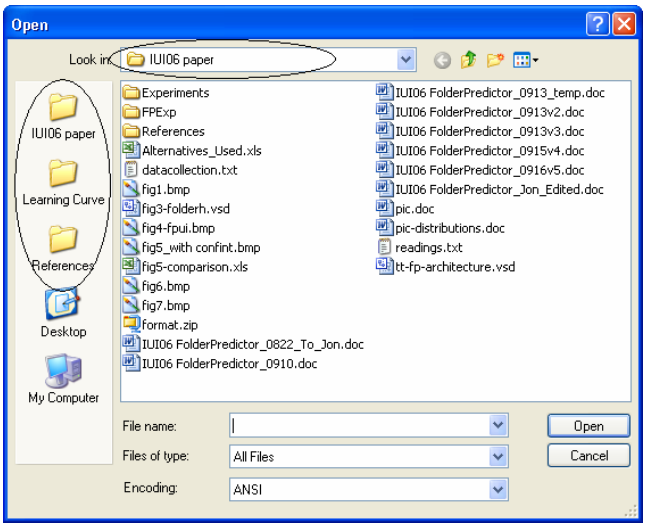

Figure 4. FolderPredictor is integrated into the open file dialog box.

There are five slots in the places bar. By default, Microsoft Windows places five system folders (including "My Computer" and "Desktop") there. Informal questioning of Windows users revealed that several of these shortcuts were not commonly used.

Thus, we felt it was safe to replace some of them by predicted folders. By default, FolderPredictor uses three slots for predicted folders and leaves two system folders. This behavior can be configured by the users if they want to see more system folders in the places bar.

Microsoft Office uses a file dialog from a custom library, while most other Windows applications use the file dialog from the common Win32 dialog library provided by the operating system. Thus, FolderPredictor needs two separate sets of instrumentation for the two types of file dialog.

6.1 Instrumentation for the common Win32 file dialog

In order to modify the places bar in the common Win32 dialog, FolderPredictor creates five entries named "Place0" to "Place4" under the registry key "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\ComDlg32\Placesbar\". These five entries correspond to the five slots, and their values can be set to the paths of the predicted folders or to the system-independent numeric IDs (CSIDL) of the system folders.

Modifying the default folder is much more difficult. There is no documented support for this feature from Microsoft. FolderPredictor modifies the default folder by injecting an add-in (a .DLL file written in C with embedded assembly language) to all applications when they are started. This add-in intercepts Win32 API calls invoked by the injected application. Common Win32 applications show the Open/Save file dialog by invoking API calls named "GetOpenFileName" or "GetSaveFileName" with the default folder passed as a parameter. Thus, the add-in can modify the default folder by intercepting the API call, changing the parameter, and then passing it on. A detailed introduction of API interception technology can be found in [13].

By intercepting the above two API calls, we can also get the original default folder of the file dialog and the folder returned by the file dialog. These folders were used in the evaluation of FolderPredictor, which is presented in the next section.

6.2 Instrumentation for the Microsoft Office file dialog

As with the common Win32 file dialog, the places bar in the Microsoft Office file dialog can be modified by manipulating registry keys. The pertinent registry key is "HKEY_CURRENT_USER\Software\Microsoft\Office*VersionNumber*\Common\Open Find\Places\", inside which *VersionNumber* should be replaced by the version number of Microsoft Office software installed on the computer.

Microsoft Office uses a file dialog from a custom library, and the API calls in this library are not exposed. Therefore, API interception technology can not be used to modify the default folder of Microsoft Office file dialog. FolderPredictor hooks into the Microsoft Office file dialog by loading macro files into Microsoft Office applications, e.g. .DOT file for WORD and .XLA file for EXCEL. Code in these macro files, written in Visual Basic for Applications (VBA), is invoked when the file dialog is called. When invoked, the VBA code changes the default folder of the file dialog to the predicted folder and then shows the dialog box.

The original default folder and the folder returned by the file dialog are also recorded for evaluation.

7. EXPERIMENTS

In this section, we evaluate the performance of FolderPredictor.

7.1 Setup

The data sets for the evaluation came from four different users of FolderPredictor. Each data set is a list of predictions that FolderPredictor has made for a user, ordered by time. Each prediction is marked with the name of the task that was active at the moment this prediction was made. The information about these data sets is shown in Table 1. The size of a data set is the number of predictions it contains.

Table 1. Information about the data sets on which experiments were conducted

#	User Type	Data Collection Time	Set Size
1	Professor	12 months	1748
2	Professor	4 months	506
3	Graduate Student	7 months	577
4	Graduate Student	6 months	397

The discount factor α is set to 0.85 for all users.

7.2 Average cost

Figure 5 compares the average cost of FolderPredictor and *Windows Default* on all four data sets. The cost of Windows Default is the distance between the original default folder of the file dialog and the destination folder. In other words, the cost of Windows Default is the user cost when FolderPredictor is not running. The figure also shows 95% confidence intervals for the costs.

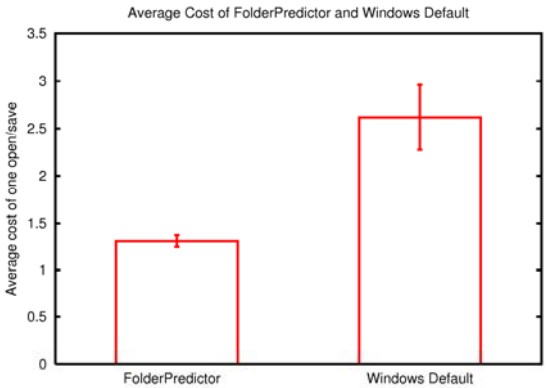

Figure 5. Average cost of FolderPredictor and Windows Default.

Statistical significance testing shows that FolderPredictor surely reduces the user cost of locating files (P-value = 1.51E-29 in

ANOVA). On average, the cost is reduced by 49.9% when using FolderPredictor.

7.3 Distribution of costs

Figure 6 shows the distribution of different costs on all four data sets, for both FolderPredictor and Windows Default. We can see from the figure that

1) About 90% of the FolderPredictor's costs are less than three. Only a small fraction of the FolderPredictor's costs are very high.

2) Although about half of the Windows Default's costs are zero, about 40% of the Windows Default's costs are above three.

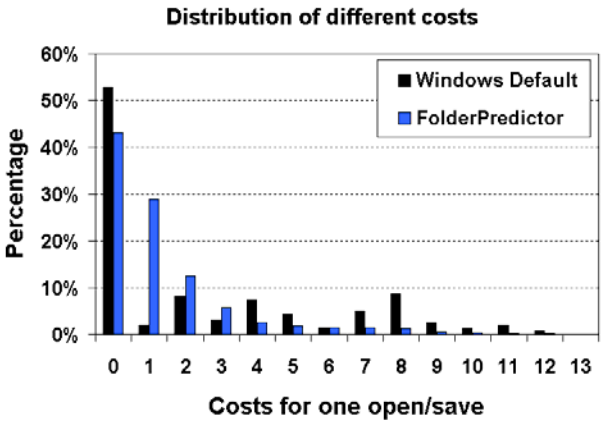

Figure 6. Distribution of costs.

This means that FolderPredictor not only reduces the overall average cost of locating files, but also decreases the probability of encountering very high costs in locating files.

It is interesting to see that Windows Default actually gets the default folder perfectly correct more than FolderPredictor. This most likely happens because Windows typically picks a leaf folder (i.e., the most recently-used folder) as the default folder. FolderPredictor sometimes plays it safe and picks an ancestor folder that is more likely to be close to multiple possible folders, and less likely to be perfect. Thus we see a large number of cases relative to the windows default where FolderPredictor is one, two, or three clicks away.

7.4 Learning curve

Machine learning systems usually perform better as more training data is acquired. In FolderPredictor's case, the training data are the user's opens and saves per task. For each open/save, FolderPredictor makes a prediction and uses the feedback from the user to update itself. Therefore, the cost of the folders recommended by FolderPredictor should decrease as more predictions are made for a task. To demonstrate this, we present the learning curve for FolderPredictor in Figure 7.

In the figure, the X-axis is the number of predictions within one task aggregated into ranges of width 10, and the Y-axis is the average cost of one prediction within this range. For example, the first point of the curve shows the average cost of all predictions between (and including) the 1st and 10th predictions of all tasks. The figure also shows 95% confidence intervals for the average costs.

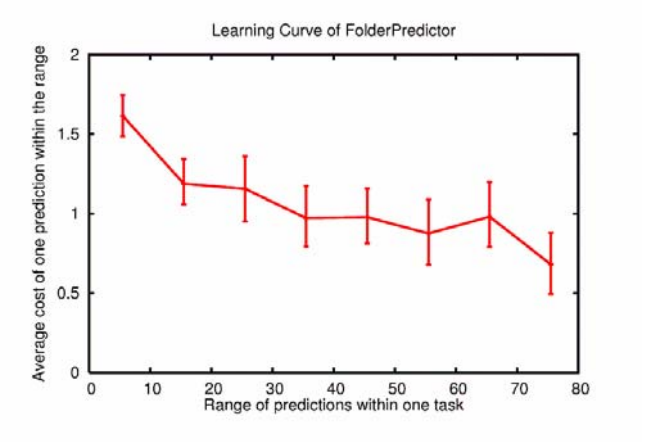

Figure 7. Learning curve of FolderPredictor.

The curve clearly shows that the cost decreases as more predictions made. The average cost decreases from 1.6 (first 10 predictions) to 0.7 (71st to 80th predictions).

8. RELATED WORK

On the surface, the FolderPredictor approach is similar to some email classification systems, which predict possible email folders for each incoming email message based on the text in the messages [14, 16, 17]. In particular, both MailCat and FolderPredictor present their top three predicted folders to the user [17]. However our approach is different in the following aspects:

1) Our predictions are made for file folders, not email folders. Predicting file folders should be much more difficult than predicting email folders. One reason is that in most situations, there are many more file folders than email folders. Furthermore, there are many different types of files under file folders, not only email messages.

2) Our predictions are based on user activities, not text in files. Text-based approaches may be applicable for email foldering, but they are more challenging to apply to folder prediction. Challenges include (a) tasks with similar keyword profiles but different folders (e.g., the class I taught last year versus the class I am teaching this year), (b) files from which it is hard to extract text, (c) ambiguity in language, and (d) computation time to extract and analyze text.

There are also some software tools that help users to quickly locate their files in the open/save file dialog, e.g. *Default Folder X* [18] for Mac and *Direct Folders* [4] for Windows. These tools make the open/save file dialog more configurable and comprehensive to the user. The user can put more shortcuts in the dialog, as well as define a default folder for each application. However, these tools cannot adjust the shortcuts and default

folders automatically based on the context of the user's activities. On the other hand, our approach makes folder predictions based on the user activities within each task. Therefore, FolderPredictor can be a good complement to these tools.

9. DISCUSSION AND CONCLUSION

To reduce user cost, we are applying machine learning to records of user activity to make useful recommendations. In this paper, we report on the FolderPredictor, which can reduce the user's cost for locating files in folder hierarchies. FolderPredictor applies a cost-sensitive online prediction algorithm to the user's activities. Experimental results show that, on average, FolderPredictor reduces the cost of locating a file by 50%. Perhaps even more importantly, FolderPredictor practically eliminates cases in which a large number of clicks are required to reach the right folder. Another advantage of FolderPredictor is that it does not require users to adapt to a new interface. Its predictions are presented directly in the open/save file dialogs. Users have nothing new to learn.

The results reported in this paper are likely a substantial underestimate of the value of the FolderPredictor. One of the key assumptions of this paper is that the user always knows which folder they want to get to and where it is located – in such a case, we have demonstrated that FolderPredictor will get them there faster. The reality is that people have limited memory and highly multitasking people often cannot maintain in their memory the locations of files on all their tasks, particularly tasks that they have not worked on recently. Thus users may need many more clicks to "search" for the right folder. By default, Windows only remembers what was worked on most recently, regardless of task. FolderPredictor on the other hand remembers multiple folders used on a task, regardless of how long ago that task was last active. Thus FolderPredictor's recommendations can serve as a reminder of where files related to a task have been stored.

There are other psychological benefits of FolderPredictor that are harder to evaluate. For example, being placed consistently in the wrong folder can generate frustration, even if the click distance is not far. At this stage, all that we have is qualitative evidence of this. During the deployments of FolderPredictor, multiple subjects reported becoming "addicted" to FolderPredictor – they were distressed when they had to use computers that did not have TaskTracer installed. They also reported that they did a better job of notifying TaskTracer of task switches in order to ensure that the FolderPredictor recommendations were appropriate for the current task.

10. ACKNOWLEDGMENTS

This project was supported in part by the National Science Foundation under grant IIS-0133994 and by the Defense Advance Research Projects Agency under grant no. HR0011-04-1-0005 and contract no. NBCHD030010. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the Defense Advanced Research Projects Agency, or the Department of Interior-National Business Center.

11. REFERENCES

[1] Barreau, D.K. and Nardi, B. Finding and reminding: File organization from the desktop. *ACM SIGCHI Bulletin*, 27 (3), 39-43, 1995.

[2] Boardman, R and Sasse, M.A. "Stuff goes into the computer and doesn't come out": A Cross-tool Study of Personal Information Management, In *Proceedings of CHI 2004, ACM Conference on Human Factors in Computing Systems*, CHI Letters 6(1), 2004.

[3] Budzik, J., and Hammond, K. J. User Interactions with Everyday Applications as Context for Just-in-Time Information Access. *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, ACM Press, 2000.

[4] Code Sector Inc. Direct Folders, http://www.codesector.com/directfolders.asp

[5] Dourish, P., Edwards, W. K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D. B., Thornton, J. Extending document management systems with user-specific active properties. *ACM Transactions on Information Systems* 18(2): 140-170, 2000.

[6] Dragunov, A., Dietterich, T. G., Johnsrude, K., McLaughin, M., Li, L. and Herlocker, J. L. Tasktracer: A Desktop Environment to Support Multi-Tasking Knowledge Workers. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pp. 75-82, 2005.

[7] Dumais, S. et al. Stuff I've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th Annual International ACM SIGIR conference on Research and Development in Informaion Retrieval (SIGIR 2003)*, pp. 72-79, 2003.

[8] Horvitz, E., Breese, J., Heckerman, D., Hovel, D. and Rommelse, K. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, July 1998.

[9] Jones, W., Phuwanartnurak, A. J., Gill, R. and Bruce, H. Don't Take My Folders Away! Organizing Personal Information to Get Things Done. In *CHI '05 extended abstracts on Human factors in computing systems*, ACM Press (2005), 1505-1508.

[10] Jul, S. and Furnas, G. W. Navigation in electronic worlds: Workshop report. *ACM SIGCHI Bulletin*, 29(2):44-49, 1997.

[11] Ko, A., Aung, H. H., and Myers, B. Eliciting design requirements for maintenance-oriented IDEs: A detailed study of corrective and perfective maintenance tasks, *International Conference on Software Engineering*, St. Louis, MO, pp. 126-135, May 15-21, 2005.

[12] Maes, P, Agents that reduce work and information overload. *Communications of the ACM*, 37(7): 30 – 40, 1994.

[13] Pietrek, M. *Windows 95 System Programming Secrets*. IDG Books Worldwide, Inc., Foster City, CA 94404, USA, ISBN 1-56884-318-6, 1995.

[14] Rennie, J. ifile: An application of machine learning to e-mail filtering. In *Proc. KDD 2000 Workshop on Text Mining*, Boston, MA, 2000.

[15] Rhodes, B. Using Physical Context for Just-in-Time Information Retrieval. *IEEE Transactions on Computer*s, Vol 52, No. 8, pp. 1011-1014, 2003.

[16] Segal, R. B. and Kephart, J. O. Incremental learning in SwiftFile. In *Proceedings of the 2000 International Conference on Machine Learning.* Morgan Kaufmann, San Francisco, 2000.

[17] Segal, R. B. and Kephart, J. O. Mailcat: An intelligent assistant for organizing e-mail. In *Proceedings of the Third International Conference on Autonomous Agents*, 1999.

[18] St. Clair Software. Default Folder X, http://www.stclairsoft.com/DefaultFolderX/

[19] Stumpf, S., Bao, X., Dragunov, A., Dietterich, T. G., Herlocker, J., Johnsrude, K., Li, L., Shen, J. Predicting User Tasks: I Know What You're Doing! *20th National Conference on Artificial Intelligence (AAAI-05), Workshop on Human Comprehensible Machine Learning*, Pittsburgh, PA, 2005.

[20] Teevan, J., Alvarado, C., Ackerman, M.S. and Karger, D.R. The Perfect Search Engine Is Not Enough: A Study of Orienteering Behavior in Directed Search. In *the ACM Conference on Human Factors in Computing Systems (CHI '04)*, (Vienna, Austria, 2004).