Understanding and Improving Personal File Retrieval

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Doctor of Philosophy
in the
University of Canterbury
by
Stephen Fitchett

Supervision and Examination Committee

Professor Andy Cockburn	Senior Supervisor
Professor Tim Bell	Associate Supervisor
Professor Mark Apperley	Internal Examiner
Dr. Ofer Bergman	External Examiner

Department of Computer Science and Software Engineering
University of Canterbury
2013

Abstract

Personal file retrieval – the task of locating and opening files on a computer – is a common task for all computer users. A range of interfaces are available to assist users in retrieving files, such as navigation within a file browser, search interfaces and recent items lists. This thesis examines two broad goals in file retrieval: *understanding* current file retrieval behaviour, and *improving* file retrieval by designing improved user interfaces.

A thorough understanding of current file retrieval behaviour is important to the design of any improved retrieval tools, however there has been surprisingly little research about the ways in which users interact with common file retrieval tools. To address this, this thesis describes a longitudinal field study that logs participants' file retrieval behaviour across a range of methods, using a specially developed logging tool called FileMonitor. Results confirm findings from previous research that search is used as a method of last resort, while providing new results characterising file retrieval. These include analyses of revisitation behaviour, file browser window reuse, and interactions between retrieval methods, as well as detailed characterisations of the use of navigation and search.

Knowledge gained from this study assists in the design of three improvements to file navigation: Icon Highlights, Search Directed Navigation and Hover Menus. *Icon Highlights* highlight items that are considered the most likely to be accessed next. These highlights are determined using a new algorithm, AccessRank, which is designed to produce a set of results that is both accurate and stable over time. *Search Directed Navigation* highlights items that match, or contain items that match, a filename search query, allowing users to rehearse the mechanisms for expert performance in order to aid future retrievals, and providing greater context than the results of a traditional search interface. *Hover Menus* appear when hovering the mouse cursor above a folder, and provide shortcuts to highly ranked files and folders located at any depth within the folder. This allows users to reduce navigation times by skipping levels of the file hierarchy.

These interfaces are evaluated in lab and field studies, allowing for both precise analysis of their relative strengths and weaknesses, while also provid-

ing a high degree of external validity. Results of the lab study show that all three techniques reduce retrieval times and are subjectively preferred by participants. For the field study, fully functional versions of Icon Highlights and Search Directed Navigation are implemented as part of Finder Highlights, a plugin to OS X's file manager. Results indicate that Icon Highlights significantly reduce file retrieval times, and that Search Directed Navigation was useful to those who used it, but faces barriers to adoption.

Key contributions of this thesis include a review of previous literature on file management, a thorough characterisation of file retrieval behaviour, improved algorithms for predicting user behaviour and three improved interfaces for file retrieval. This research has the potential to improve a tedious activity that users perform many times a day, while also providing generalisable algorithms and interface concepts that are applicable to a wide range of interfaces beyond file management.

Publications Arising from this Thesis

The peer-reviewed publications listed below are based on work completed as part of this thesis. Chapters on which they are based are noted in brackets.

1. **Fitchett, S.** and Cockburn, A. AccessRank: Predicting What Users Will Do Next. In *CHI '12: SIGCHI Conference on Human Factors in Computer Systems, 2012*, (Austin, Texas, USA), ACM, 2239–2242. **CHI Honorable Mention Award**. (Chapter 6).

2. **Fitchett, S.**, Cockburn, A., and Gutwin, C. Improving Navigation-Based File Retrieval. In *CHI '13: SIGCHI Conference on Human Factors in Computer Systems, 2013*, (Paris, France), ACM, 2329–2338. **CHI Best Paper Award**. (Chapter 7).

3. **Fitchett, S.**, Cockburn, A, and Gutwin, C. Finder Highlights: Field Evaluation and Design of an Augmented File Browser. In *CHI '14: SIGCHI Conference on Human Factors in Computer Systems, 2014*, (Toronto, Canada), ACM. To appear. **CHI Honorable Mention Award**. (Chapter 8).

Additionally, the following publications are based on work conducted during my doctoral studies, but not directly part of the thesis. The second is the result of a three month internship at Microsoft Research Asia in late 2010.

4. Cockburn, A., Quinn, P., **Fitchett, S.** and Gutwin, C. Improving Scrolling Devices with Document-Length-Dependent Gain. In *CHI '12: SIGCHI Conference on Human Factors in Computer Systems, 2012*, (Austin, Texas, USA), ACM, 267–276.

5. Edge, D., **Fitchett, S.**, Whitney, M. and Landay, J.. MemReflex: Adaptive Flashcards for Mobile Microlearning. In *MobileHCI '12: Proceedings of ACM MobileHCI '12, 2012*, (San Francisco, CA, USA), ACM, 431–440. **MobileHCI Best Paper Award**.

Technical Acknowledgements

The work contained in this thesis is my own, except where stated otherwise. My senior supervisor, Andy Cockburn, provided inspiration, ideas, and proofreading, and his contributions are weaved throughout the thesis.

Publications 1 to 3 were all written in collaboration with Andy Cockburn. While they were primarily my work, he contributed to the ideas and editing of all the papers.

The research for Chapter 7 was substantially done at *The Interaction Lab* at the University of Saskatchewan, and Professor Carl Gutwin provided much of the inspiration for this work. He was also involved in editing publications 2 and 3. The study in Chapter 7 was conducted in The Interaction Lab by Roxanne Dowd and Jared Cechanowicz.

My thanks go to all those who were involved in research collaborations that contributed to this thesis.

Ethical Considerations

The studies performed as part of this thesis involved human participants. Every care was taken to ensure their privacy and comfort was maintained at all times. Participants maintained the right to withdraw their participation or data at any point.

The studies undertaken in Chapters 5 and 8 are covered by the University of Canterbury's Human Ethics Approval application HEC2012/06. The study in Chapter 7 was undertaken at *The Interaction Lab* at the University of Saskatchewan, and was covered by Professor Carl Gutwin's ethics approval.

All participants signed paper consent forms or agreed to online consent forms before any log data, demographic information or survey responses were collected. Copies of these forms are reproduced in the appendices.

Acknowledgements

I would like to thank the following people and groups for their support during this thesis.

First, thanks to my supervisor, Professor Andy Cockburn, for his invaluable guidance, support, inspiration, wisdom, patience and advice throughout my research, without which this thesis would not exist. Thanks also to my associate supervisor, Professor Tim Bell, for his guidance and always-positive attitude.

Thanks to Professor Carl Gutwin at the University of Saskatchewan, who kindly hosted me in his lab for several months in 2012 and provided valuable guidance during that time. Thanks also to the faculty and students who hosted me as part of brief visits to the University of Calgary and the University of British Columbia.

Thank you to my colleagues in the HCI lab (Joey, Philip, Sylvain, Mathieu and Joshua) for the lively discussions, and for providing me with some much-needed human contact at some of the busier points of my PhD. Thanks also to the staff and other postgraduate students in the computer science department for all the discussions and assistance over the years.

Thanks to my friends for having more confidence in my ability to finish my thesis than I did, and for understanding when they did not see me for long periods.

Finally, thanks to all my anonymous participants, without which the studies in this thesis would not have been possible, for volunteering their time and data.

This thesis was financially supported by the *Canterbury Scholarship*, awarded to the top eight applicants for a University of Canterbury Doctoral Scholarship each year. Additional funding was provided by the *New Zealand Royal Society Marsden Grant 10-UOC-020*. Visits to labs in Canada were financially supported by the *Natural Sciences and Engineering Research Council of Canada* and the *GRAND NCE*.

Table of Contents

List of Figures

List of Tables

Chapter I

Introduction

Retrieving files is an extremely common tasks for all computer users, performed many times a day [102]. Accordingly, a variety of techniques have been developed to assist with this task, including sophisticated search tools and 'Open Recent' menus. However, navigating to files by traversing through a file hierarchy in a file browser remains the most common method to retrieve files [25]. Nevertheless, navigation is often slow, with retrievals taking an average of 12-17 seconds each [30].

While the tools available to retrieve files have advanced, file hierarchies have become larger. As an example, the original Macintosh File System, introduced in 1984, supported a maximum of 4094 files and two hierarchy levels, while its successor, Hierarchical File System (HFS), supported 65535 files. In comparison, modern personal computers commonly hold millions of files. While a considerable proportion of these files are system files or application data that users do not normally interact with, the increasing hierarchy size nevertheless increases the importance of efficient retrieval mechanisms.

Advances in the user interfaces used to retrieve files have provided some assistance to users. Search tools have become faster and more sophisticated, while other features facilitate access to recent and frequent files. Improvements have also been made to file browsers, which is particularly important given user preferences towards navigation. For example, aliases, shortcuts and symbolic links allow users to create multiple paths to a single location. Many file browsers allow users to apply tags or labels to files. More recently, features such as virtual folders and saved searches present dynamic content based on pre-determined rules. However, all these file browser features require organisational effort in advance, while research has shown that users dislike manual organisational strategies [155, 52]. Difficulties in imagining

future retrieval requirements can also make organisation overwhelming [185].

Considering the frequency with which people retrieve files, any improvements in retrieval times have the potential to be hugely beneficial. However, surprisingly little is known about how users retrieve their files and why they do so in a particular way. Such understanding is important to the design of any new or improved techniques that aim to reduce file retrieval times.

This thesis is therefore focused on *understanding and improving personal file retrieval*. This objective is split into two parts: first, understanding how people currently retrieve their files, and second, creating new interfaces (including their underlying algorithms) to assist users in retrieving their files more efficiently.

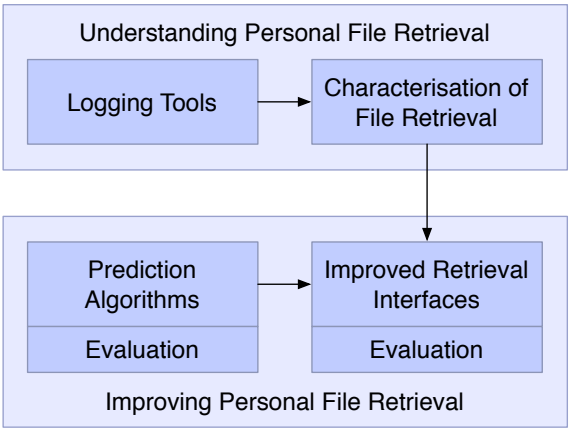


Figure 1.1: Components of this thesis

These goals are illustrated in Figure 1.1. The first goal, understanding personal file retrieval, was achieved with the creation of a logging tool that unobtrusively monitors file retrieval behaviour. This logging tool was installed on 26 participants' computers for four weeks, and analysis of the logs produced by this study, combined with interview responses, allowed for a comprehensive characterisation of personal file retrieval. This characterisation helps to answer important questions such as when and why users choose to use particular retrieval methods, as well as potential areas for improvement. The second goal, improving personal file retrieval, was achieved by building on the knowledge gained from this characterisation to provide three new techniques that augment the standard navigation interface to facilitate

faster file retrievals: Icon Highlights, Search Directed Navigation and Hover Menus. These techniques also use advanced back-end predictive algorithms described as part of this thesis.

The remainder of this chapter formally defines the research goals, methods and primary research contributions of this thesis, followed by an outline of the content of the thesis. It concludes with a glossary of common terms used throughout the thesis.

1.1 Problem Statement

The two primary goals of this thesis are to understand and improve personal file retrieval. These two goals are interlinked: in order to *improve* file retrieval tools, it is important to first *understand* how users currently retrieve their files and the deficiencies in current methods.

The two primary goals are further broken down into the following subgoals that aim to understand and then improve personal file retrieval. Criteria for judging their successful completion are also provided.

1. Understand the methods with which users can currently retrieve files, and understand retrieval behaviour in the context of existing knowledge about file management. Successful completion of this goal will result in a description and categorisation of commonly available file retrieval methods and an analysis of how people use them based on existing literature.

2. Form an empirical characterisation of file retrieval behaviour based on how users retrieve their files in a natural setting. Successful completion of this goal will result in a complete description of retrieval behaviour over a range of retrieval methods, and include both precise quantitative analysis as well as qualitative comments from participants that help to frame the quantitative results.

3. Develop and evaluate a prediction algorithm that can accurately predict future user actions, including file retrievals, while also providing a relatively stable set of results to facilitate spatial stability in user

interfaces that use it. This goal will be successful if this new algorithm outperforms existing algorithms when considering both the accuracy and stability of results.

4. Using the knowledge gained from the first two subgoals, and prediction algorithms from the third subgoal, design, implement and evaluate improved file retrieval methods that allow users to retrieve their files more efficiently. Successful completion of this goal will result in retrieval methods that empirically and subjectively outperform existing methods in a natural setting.

1.2 Research Approach

File retrieval is a frequent activity that occurs both with paper files and in a range of electronic settings. This thesis is focused on electronic file retrieval performed on personal computers using a hierarchical file structure.

The literature provides a broad background of file management activities. This prior work is reviewed, along with that of other domains where users must also organise and retrieve large amounts of data, such as email management and webpage retrieval. Comparing properties of these domains, and discussing how differences in these properties affect organisation and retrieval behaviour in each, highlights the challenges that are unique to the file retrieval domain.

This work aims to further understand these challenges and describe more precisely the ways in which users retrieve their files. Previous researchers have described file retrieval by conducting interviews, observing users interacting with their files (in person or using video), using questionnaires and performing lab experiments. While all these approaches are valuable, none of them allow for precise, objective, externally valid quantitative data on retrieval behaviour: interviews provide qualitative data only; observation is time consuming, does not allow for precise measurement, and participants may act differently when they know they are being observed; questionnaires provide subjective data only; and it is difficult to ensure external validity when performing lab experiments.

To overcome this problem, a logging tool is implemented that monitors file retrieval behaviour in OS X. This tool is deployed in a four-week longitudinal study, providing precise, objective quantitative data on file retrieval. Participants reported that they quickly forgot it was running, suggesting a high degree of external validity.

The main limitation of logging tools is that they only report *what* users have done, not *why* they did it. To ease this limitation, the file retrieval logs are supplemented by open-ended interviews that allow users to put their file retrieval behaviour in context.

The data gained from this improved understanding of file retrieval behaviour helps to shape the design of new file retrieval interfaces. These interfaces rely on new prediction algorithms, which are evaluated and calibrated using simulations. By using real user interaction logs from a variety of domains, these simulations provide a fast and accurate way of quantitatively assessing the algorithms' predictive power.

Three new file retrieval interfaces are developed, which are initially evaluated with a lab study that uses simulated file retrieval tasks to assess their potential performance and the relative benefits of each interface. This allows for precise analysis of their strengths and weaknesses, such as the differing effect of spatial stability and visit counts on retrieval times for each interface.

Although the tasks and experimental system are designed to approximate real world retrieval behaviour, for example by using similar folder sizes and retrieval depths as have been found in previous research, there are still substantial differences between simulated file retrievals in a lab setting and real-world file retrieval behaviour. The potential performance improvements shown by the lab study are therefore further tested by the development of Finder Highlights, which includes fully functional versions of two of the new retrieval interfaces as part of a popular file browser. Finder Highlights is then evaluated in a four-week longitudinal field study, demonstrating how the interfaces are used in natural settings. The combination of lab and field studies therefore illustrates both the precise strengths and weaknesses of each interface, as well as whether they are successful overall at improving file retrieval. Both the lab and field studies are complemented with surveys, providing subjective feedback from participants.

1.3 Research Contributions

This thesis makes six primary contributions to the research knowledge of personal file retrieval. These are:

1. A review of file retrieval methods and file management. This review allows other researchers to more quickly assess current retrieval methods and opportunities for further research.

2. The development of FileMonitor, a tool that logs file retrieval behaviour on OS X. FileMonitor enables accurate and automated collection of large amounts of file retrieval data, enabling researchers to characterise file retrieval or collect data with which to evaluate file retrieval interfaces or prediction algorithms. Previously, file retrieval research was limited to other techniques such as surveys, lab studies or video analysis, which provides a more limited view of overall file retrieval behaviour. The description of FileMonitor's implementation is also of use for researchers developing similar logging tools.

3. An empirical characterisation of file retrieval behaviour. FileMonitor was installed on the personal computers of 26 participants for a four week period, followed by 30 minute interviews about their file retrieval and organisation behaviour. FileMonitor logs and interview responses provide a detailed characterisation of file retrieval behaviour, with specific analyses of file revisitation, filename content, hierarchy depth, interactions between retrieval methods, use of different file browser views, use of navigation features, file browser window reuse, search behaviour, use of other retrieval methods, and file management tasks. Findings from this characterisation have implications for both the design and evaluation of file retrieval tools.

4. The development and evaluation of AccessRank, a prediction algorithm designed for use in a range of user interfaces. AccessRank is designed with both the accuracy and stability of results in mind, with stability

particularly important in user interfaces where spatially stable interfaces allow users to utilise their spatial memory to improve performance. Simulations on a range of datasets compare various configurations of AccessRank against existing algorithms, showing that AccessRank outperforms existing algorithms when considering the combination of accuracy and stability, and also demonstrating the strengths of different algorithms across different datasets.

5. The design and evaluation of three prototype interfaces that augment a standard file browser to aid file retrieval tasks: Icon Highlights, Search Directed Navigation, and Hover Menus. Icon Highlights facilitate visual search by highlighting items that are likely to be accessed next, using the AccessRank prediction algorithm. Hover Menus allow users to reduce the number of navigation steps to reach target items by providing a menu containing shortcuts to likely files and folders within a particular folder. Search Directed Navigation highlights items that match, or contain items that match a filename query, bridging the gap between navigation and search. A lab study of 16 participants shows that each improve file retrieval performance and are subjectively preferred over a standard browser.

6. The design and evaluation of fully functional implementations of Icon Highlights and Search Directed Navigation, implemented as part of a plugin to a real-world file browser and evaluated in a four-week field study with 19 participants. The implementations include design and algorithmic enhancements to enable the techniques to work in a real-world setting, including a specialised version of AccessRank to predict future file retrievals for Icon Highlights, and a search algorithm to determine filename matches for Search Directed Navigation. Results of a four-week field study confirm the performance benefits provided by Icon Highlights and highlight the advantages that field studies can offer over lab studies.

1.4 Structure of the Thesis

The thesis is divided into five parts: a review of the current knowledge about file organisation and retrieval; an empirical characterisation of file retrieval; the development of prediction algorithms for user interfaces; new interfaces to improve navigation-based file retrieval; and a discussion of the topics raised in this thesis and directions for future work.

To begin, Part I provides an overview of existing knowledge of file retrieval. Chapter 2 describes current file retrieval methods, including navigation, search, and recent items interfaces, and discusses the properties of each. The chapter has a particular focus on widely used commercial systems. Chapter 3 takes a more user-centric focus on file management, reviewing existing literature on how users organise their hierarchies and use the retrieval methods described in the preceding chapter. It also describes how file management differs from similar domains, such as email and bookmark management, and discusses how these differences affect relative use of different retrieval methods.

Part II is focused on characterising current file retrieval behaviour. Understanding how users currently retrieve their files is an important first step to designing any improved file retrieval tool. This research is divided into two parts, described below: the first describes the FileMonitor logging tool, a tool that monitors file retrieval behaviour, and the second describes the results of a four-week field study using FileMonitor, providing a comprehensive characterisation of file retrieval.

FileMonitor is described in Chapter 4. FileMonitor runs on OS X, and silently monitors usage information about three types of interaction: Finder usage, describing detailed interaction within OS X's primary file browser, such as opening folders, renaming files, and creating new windows; Spotlight usage, detailing interaction with OS X's system-wide search menu, including query changes and results selections; and external file retrievals, providing the paths of all files opened with any method not directly observed (such as 'Open Recent' menus). The chapter describes the implementation details of FileMonitor, as well as comprehensive details about the content of FileMonitor log files.

Chapter 5 describes the results of a four-week field study in which FileMonitor was deployed. Participants ran FileMonitor on their personal computers and took part in 30 minute interviews following the study. Log analysis, combined with participant comments from interviews, provides a detailed characterisation of file retrieval. Broadly, the characterisation includes results about the files that participants retrieved, such as revisitation patterns, file types, and hierarchy depths, as well as results about the methods used to retrieve files, including reasons for using particular methods, interaction between retrieval techniques, and in depth analysis of file navigation within the file browser.

Part III steps back from file retrieval to examine the larger topic of predicting future user interactions based on their usage history. While relevant to predicting future file retrievals, the focus is on predictions that are not tied to a specific domain. Chapter 6 describes AccessRank, a new prediction algorithm designed for use in user interfaces, where accuracy and the stability of results over time are both important. The chapter first summarises existing prediction algorithms, then describes AccessRank, a framework for evaluating prediction algorithms for use in user interfaces, and the results of a simulation showing that AccessRank produces a superior combination of accuracy and stability in a range of datasets.

Part IV is focused on the second goal of the thesis – improving personal file retrieval. Building on findings from Parts II and III, it introduces new file navigation augmentations to reduce file retrieval times.

Chapter 7 introduces three improved file navigation techniques: Icon Highlights, Hover Menus and Search Directed Navigation. An initial lab study demonstrates that the techniques have the potential to substantially reduce file retrieval times, particularly when the file browser provides a low degree of spatial stability.

This work is extended in Chapter 8 to investigate whether the potential improvements found in the initial lab study are realised in real-world retrieval tasks. The chapter begins by describing Finder Highlights, a plugin for the OS X Finder that adds support for Icon Highlights and Search Directed Navigation. Finder Highlights includes refined interface designs and back-end algorithms for the two techniques. In particular, AccessRank is extended

to better support hierarchical file data for Icon Highlights predictions, and a new search algorithm is described to support quick highlighting for use with Search Directed Navigation. The chapter also introduces Benefit Weighted AccessRank to support predictions for Hover Menus, although Hover Menus is not itself implemented as part of Finder Highlights. The chapter concludes by reporting the results of a four-week field study that confirms the effectiveness of Icon Highlights and Search Directed Navigation, but finds poor adoption rates for Search Directed Navigation. The study also highlights the potential advantages that field studies offer over lab studies, suggesting that greater use of field studies in HCI would be beneficial.

Part V discusses the broader findings of the research presented in this thesis and provides direction for future work.

1.5 Glossary of Terms

While terms are rarely strictly defined in the field of human computer interaction, this section outlines the definitions used within this thesis.

1.5.1 Files

A *file* is a collection of data with a specific location in the file system, and with a specified filename. A *document* is normally associated with a single file, and can be created, edited, or viewed in an applicable application. Other types of files include, for example, applications and system files.

A *folder* is a container in the file system which can contain other files or folders. A folder may also be referred to as a *directory*. More generically, an *item* in the file system may refer to either a file or a folder.

1.5.2 File structure

A *file hierarchy* refers to the structure of the file system. The file hierarchy may be represented as a *tree*, with the root of the file hierarchy corresponding to the root of the tree. The *nodes* in the tree are the files and folders in the file hierarchy, with leaf nodes corresponding to the files. Files or folders may also be referred to as nodes in the context of other data structures, such as queues.

Portions of the file hierarchy, rooted at a particular folder, are referred to as either a *subtree* or *sub-hierarchy*.

1.5.3 Opening files

A *retrieval* is any direct user action that results in a file being opened, for example double clicking its icon in a file browser, selecting it from a menu of recent items, or launching an application. An application opening its own datafile in the background does not count as a retrieval, since it is not in response to a direct user action.

An *access* is an equivalent term to a retrieval, but is domain independent. Where the term *retrieval* is used for only some domains such as personal files, an access may also refer to a web page visit in the domain of web browsers, a window switch in the domain of window management, or any other use of an item in other domains. An access may also be referred to as a *visit*. A *revisitation* refers to accessing an item that has previously been accessed.

1.5.4 Retrieval methods

Navigation or *browsing* refer to any movement through the file hierarchy by traversing through a series of folders. This is typically performed through a *file browser*, such as *File Explorer* on Microsoft Windows or the *Finder* on OS X. *Search* refers to an interface that requires a text query before showing any candidate items.

These retrieval methods, and others, are described in detail in Chapter 2.

1.5.5 Scores

An algorithm may product numerical data for items for use for comparison purposes. This may be either a *score* or a *weight*, however a weight is normally used for a preliminary or partial calculation that leads, or can lead, to the calculation of a final score.

Part I

File Retrieval and Retrieval Tools

Chapter II

File Retrieval Methods

Files can be accessed in a range of contexts: looking for a photo taken several years ago, opening a document from an active project at the start of a day, or retrieving a previous version of a currently open document, to give a few examples. Unsurprisingly, considering these diverse contexts and the importance of file retrieval, there are a range of different file retrieval methods offered by modern operating systems.

To provide context to this thesis, this chapter describes the kinds of retrieval methods commonly available to users, with examples of specific implementations for each. It begins by providing a classification of retrieval methods. This is followed by detailed descriptions of each method. Although reference is made to some research systems, the chapter is predominately focused on techniques deployed in commercial systems – file retrieval literature is more extensively reviewed in Chapter 3.

2.1 Classification of Retrieval Methods

For the purposes of this thesis, file retrieval is defined as any process which results in accessing a document, application, or other file on a computer in response to an explicit user request. Table 2.1 categorises file retrieval methods at a high level based on two properties: whether content is presented in a hierarchical fashion, and whether the underlying content is static, automatically generated based on algorithms or defined properties, or generated based on a specific query or other user input.

File navigation – the act of traversing through the file hierarchy to reach a target file – is the most frequently used retrieval method [25]. Although the hierarchy changes over time as files and folders are created, deleted, moved

	Hierarchy	**No hierarchy**
Static content	Navigation	Bookmarks
Dynamic content (automatically generated)	Saved search	Recommender interfaces
Dynamic content (manually generated)	Faceted search	Search

Table 2.1: File retrieval methods

or renamed, the content is otherwise completely predictable, in which case the same navigation steps will yield the same files. Navigation is normally performed in a dedicated file browser, however application 'Open' dialogs typically present a similar interface.

Search interfaces provide dynamic results based on a user query and usually present results in a flat list. Modern file search interfaces allow users to search based on a large number of attributes, such as filename, content, type, author, tags or modification date. Some third party launcher tools allow users to quickly open items in response to a short input string, and are often specialised for certain types of items, such as applications. These essentially act as a specialised form of search.

Some systems allow searches to be saved as *saved searches*, where the search results are generated implicitly on each successive activation, without the need to re-enter a query. While these search results are still presented in linear form, these saved searches act like folders themselves and can be integrated with navigation as a single navigation step as part of a longer traversal.

File browsers, as well as system-wide features such as the Windows taskbar and the OS X Dock, offer *bookmarking* features to manually create quick shortcuts for applications, folders, or documents. These shortcuts provide a way to bypass the hierarchy for frequent items. In other cases, systems use algorithms to predict which items are most likely to be retrieved next. These are referred to in this thesis as *recommender interfaces*, with the most common form being the 'Open Recent' menus that are available in most applications.

Techniques that appear to be very different often have surprising similarities. For example, search and navigation are shown at opposite positions in Table 2.1. However, consider a file system where every file has a series of tags, representing their ancestor folder names. Navigation can then be seen as a highly specialised form of search where the search query is an ordered sequence of these search tags. The difference lies in the techniques' presentation methods, which have a large effect on their use cases; for example, by presenting the possible options at each step, navigation offers a crucial reminding feature [19] that removes the requirement for users to have a perfect memory of file locations, filenames or even which files exist.

The benefits of this reminding feature and the flexibility of search have inspired systems that use *faceted search* (e.g., [14]). Faceted search allows users to branch on arbitrary attribute values, providing navigation-like interfaces to an underlying search algorithm.

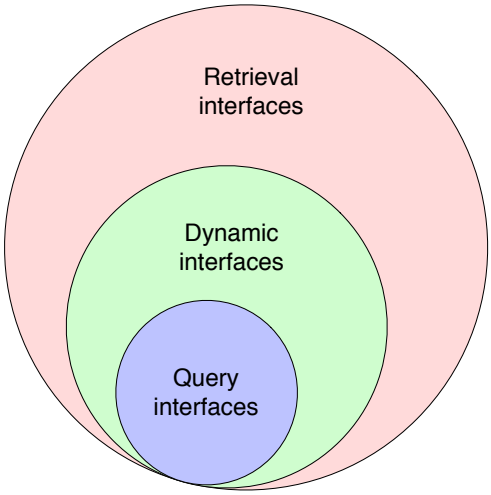


Figure 2.1: Venn diagram categorisation of retrieval methods

Figure 2.1 presents an alternate categorisation of retrieval methods, loosely based on the property of static versus dynamic content presented above. *Dynamic interfaces* include all interfaces that present a dynamically generated subset of results, in response to a set of either implicit or explicit parameters, and are a subset of retrieval interfaces. Dynamic interfaces include those such as 'Open Recent' menus (with an implicit parameter of recently

accessed items) as well as search interfaces (with explicit search parameters). Those dynamic interfaces that require an explicit query, such as search, are further defined as *query interfaces*.

As retrieval methods become more specialised in this model, the set of items they present becomes potentially more relevant and likely to have the target item readily available. However, there is a tension between specificity, accuracy and the amount of active effort required from the user to correctly configure the interface. This tension is illustrated in Figure 2.2: navigation has high accuracy (the target is always present in the hierarchy) and low cognitive effort (no explicit query is required, and navigation's reminding ability reduces cognitive load [19]) but is not specific (all files are available); search is accurate (results match a tailored search query) and specific (only items that match the query are presented) but requires more effort (to devise a suitable query); and automatic recommender interfaces such as 'Open Recent' menus are specific (small set of likely candidates) and require little effort (no query needs to be devised) but are inaccurate (as results are not tailored to a task-specific query).

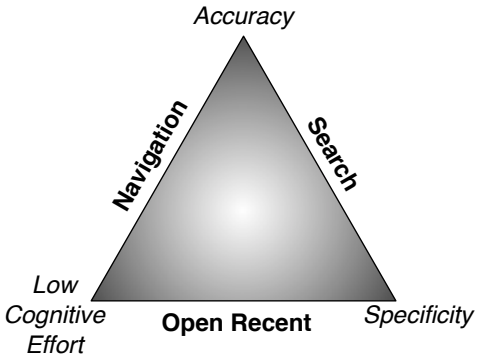


Figure 2.2: Tensions faced by retrieval methods

2.2 Support for File Retrieval Features

Modern operating systems support a diverse set of file retrieval methods and features. Table 2.2 summarises support for these in three popular operating systems (Windows 7, OS X 10.8 and Fedora 18), indicating broad similarities

between the methods available on each system. The remainder of this chapter describes details of this support, with sections on each of the main types of retrieval methods: navigation, search (including saved searches, faceted search, search-based research systems and launcher utilities), recommender interfaces and file bookmarking tools.

Retrieval feature	Windows 7	OS X 10.8	Fedora 18 w/ Cinnamon and nemo
Navigation			
Folder hierarchies	✓	✓	✓
Shortcuts	✓	✓	✓
Back and forward buttons	✓	✓	✓
Recent folders	✓(where files opened)	✓(all navigated to)	✗
Manually arrange items	✓(desktop only)	✓(desktop/icon view)	✓(desktop/icon view)
Automatically sort items	✓	✓	✓
Editable path field	✓	✗	✓
Colour labels	✗	✓	✗
Keywords	✓	✓	✗
Views			
Icon view	✓	✓	✓
List/details view	✓	✓	✓
Tree view	✓(sidebar only)	✓	✓(sidebar only)
Column view	✗	✓	✗
Dynamic locations			
Saved searches	✓	✓	✓
Multiple folders in single view	✓(libraries)	✗	✗
Previews			
In-icon preview	✓	✓	✓
Same window preview	✓	✓	✗
New window preview	✗	✓	✗
Editable preview	✗	✗	✗
Search			
Integration in file browser	✓	✓	✓
System-wide search menu	✓	✓	✓
Metadata searches	✓	✓	± (limited)
Boolean searches	✓	✓	✗
Search in open and save dialogs	✓	✓	±
Recommender interfaces			
Recent documents (per app)	✓	✓	✓
Recent documents (system-wide)	✓(not by default)	✓	✓
Recent apps (system-wide)	✗	✓	✗
Non-recency recommendations	✓(applications only)	✗	✗
Recent places in open dialogs	✓	✓	✓(save dialog only)
Bookmarks			
System-wide app bookmarks	✓(taskbar)	✓(Dock)	✓(taskbar)
System-wide doc bookmarks	± (tied to app)	✓(Dock)	✗
System-wide folder bookmarks	± (tied to Explorer)	✓(Dock)	✓(Cinnamon menu)
Bookmarks in file browser	✓	✓	✓
Bookmarks in open dialogs	✓	✓	✓

Table 2.2: Support for file retrieval features in commercial operating systems

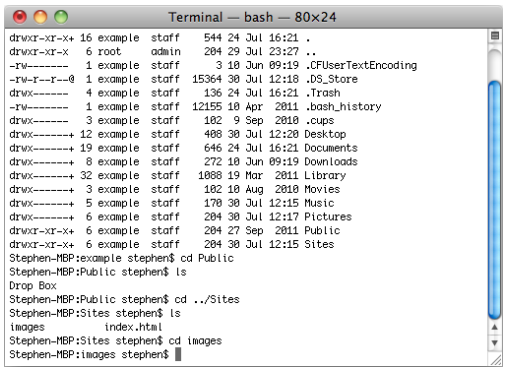
2.3 File Navigation

Navigation involves recursively opening folders in a file hierarchy until reaching a target file. Sometimes the user will have perfect recall of a target's location, and each step of the hierarchy traversal involves only scanning the view for an intermediary target and selecting it. Other times, the contents of each intermediary folder will provide reminders that assist users in reaching a target. Lansdale [121] identified two distinct psychological processes involved in navigation retrievals: recall-directed search, followed by recognition-based scanning. *Recall-directed search* involves using memory about the target item to get close to the document, while *recognition-based scanning* is the fallback option users resort to when recall fails, and uses reminders provided from recognising files and folders to help complete the retrieval.

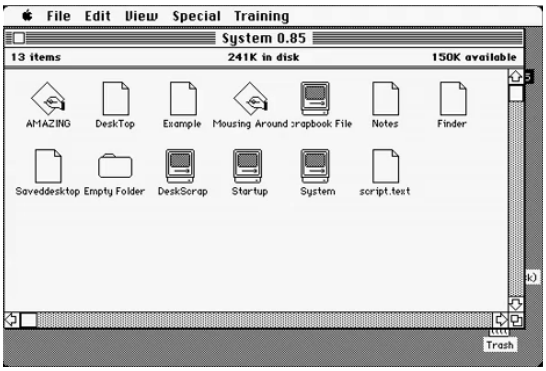
Navigation interfaces have changed substantially since their inception. Before the advent of the graphical user interface, navigation occurred only in a command line interface (Figure 2.3a). One of the first graphical file managers, Finder 1.0, is shown in Figure 2.3b. While it supported folders, they could only exist at the top level of the hierarchy, and did not appear in 'Open' dialogs (where all files were shown in a flat list).

Modern file managers provide a range of view types and navigation features. Figure 2.3c shows the Details view of Windows 7, with an expandable tree view on the left. Selecting a folder in the left pane shows its contents in the right pane. Additional features include an address field, navigation buttons and a preview of the selected item.

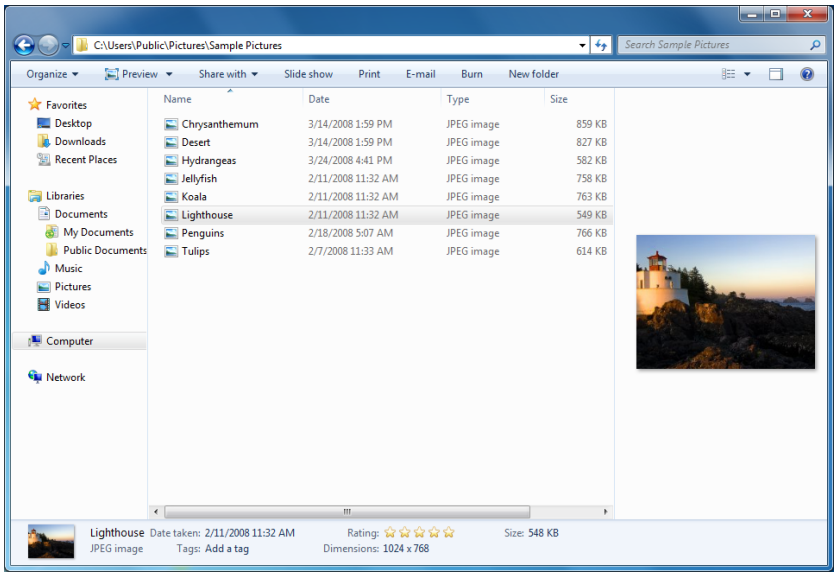
Figure 2.4 shows four view types in the Finder on OS X. *Icon view* displays items in a grid, similar to the view used in the original Finder. Items can be manually moved to custom locations, or automatically sorted, giving this view more flexibility than others. *List view* is similar to the Details view of Windows Explorer, except that folders can be expanded or collapsed like in the tree view in the left pane of Windows Explorer (Figure 2.3c). Both List view and Details view feature sortable columns for various attributes. *Column view* adds a new pane to the right for each folder that is selected, showing the contents of the selected folder. When a file is selected, an extra column shows a preview along with various file metadata. *Cover Flow view*



(a) File navigation using a command line



(b) Finder 1.0 (1983)[1]



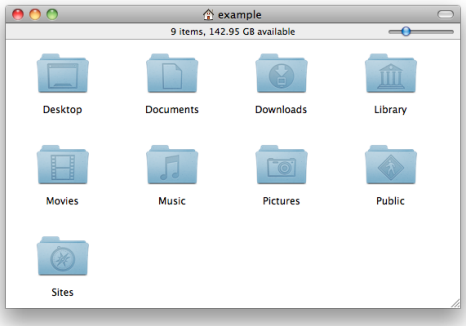
(c) Windows Explorer in Windows 7, showing folders on the left with details view on the right

Figure 2.3: File navigation interfaces

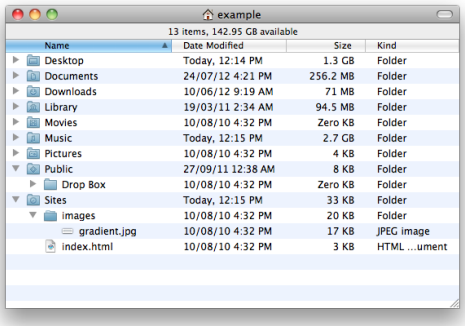
is similar to List view, but shows file previews in an extra pane at the top of the window.

The most common types of modern file managers are *spatial* and *navigational* file managers. Spatial file managers, such as Finder versions 5 to 9, use a metaphor representing files and folders as physical objects. Browser windows are tied to a particular folder; opening a folder will open a separate

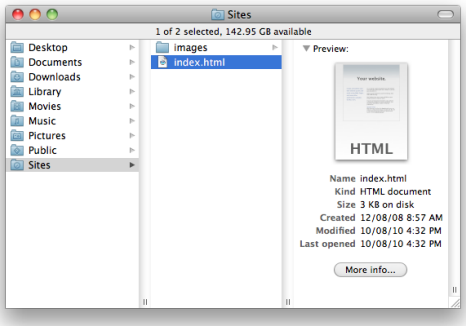
[1] Image adapted from `http://en.wikipedia.org/w/index.php?title=File:Finder10.png&oldid=467701828`.



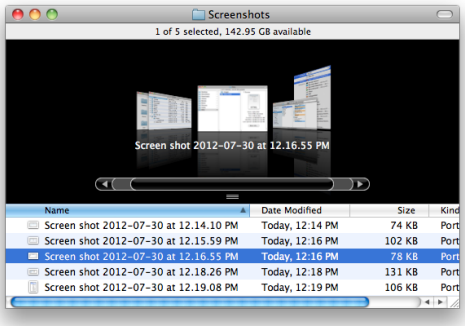
(a) Icon view



(b) List view



(c) Column view



(d) Cover Flow view

Figure 2.4: File navigation views in Mac OS X 10.6

window corresponding to that folder, and only one window can exist for each folder. Folders also preserve their spatial state, such as window position and size.

Conversely, navigational file managers, such as File Explorer on Windows and the Finder on OS X, use a navigational metaphor similar to a web browser. Users can navigate between different folders within a single browser window, and there are typically navigational controls such as back and forward buttons. Navigational file managers allow multiple windows to show the same location.

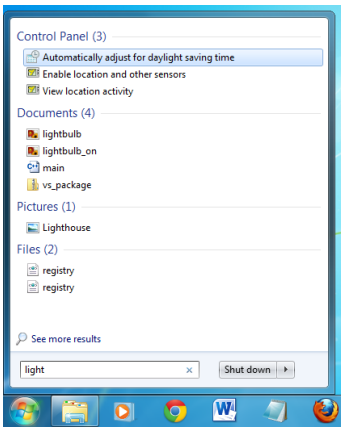
2.4 Search

Search interfaces are incorporated into most modern operating systems, such as Windows Search [139] (Figures 2.5a and 2.5b) and OS X's Spotlight [7] (Figures 2.5c and 2.5d).

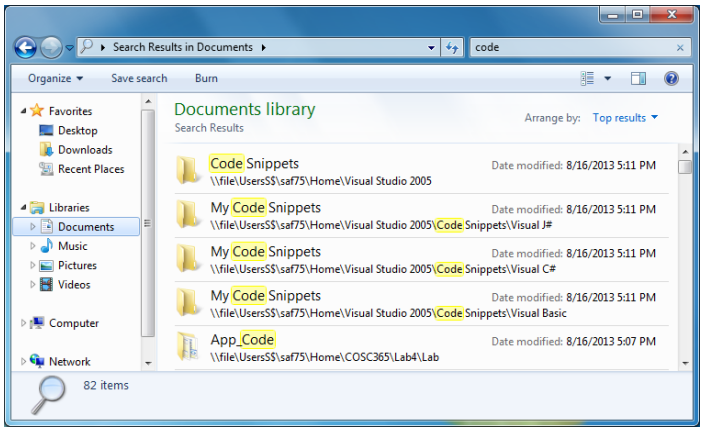
Though early search tools only searched filenames or simple attributes, many modern tools also search file content and a wide range of file metadata [162]. Current versions of both Windows and OS X provide at least two search interfaces: one showing a list of results in a menu, broken down by type and searching the entire system (Figures 2.5a and 2.5c), and another embedded in the file manager that can search specific locations and where the presentation of the results can be modified (Figures 2.5b and 2.5d).

Raskin [154] described two strategies for search interfaces: delimited search and incremental search. Delimited search involves the user specifying the entire search pattern before performing an action (such as pressing the enter key) to initiate the search. Delimited search is common in editor applications and on the internet. In contrast, incremental search systems begin searching immediately when the user starts typing, further refining the search results once more of the search pattern has been specified. Raskin identified a number of advantages of incremental search over delimited search; for example, the user need not guess how much of the pattern is required to locate their target, as they can observe the results in real time as they type. Additionally, the user will immediately know if their pattern produces no results or if it contains an error, without needing to spend more time entering a complete search pattern or waiting for a search to complete. While incremental search has traditionally been less common than delimited search, its numerous usability advantages have led to its adoption in a larger number of interfaces, and it is used with all the search interfaces in Figure 2.5.

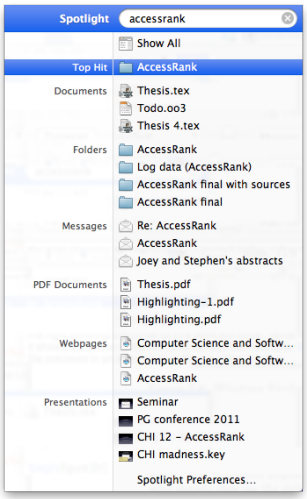
Although search is most commonly associated with interfaces like those described above, variations on search interfaces include saved searches, faceted search, launcher utilities and research systems designed as replacements for file hierarchies. These are described below.



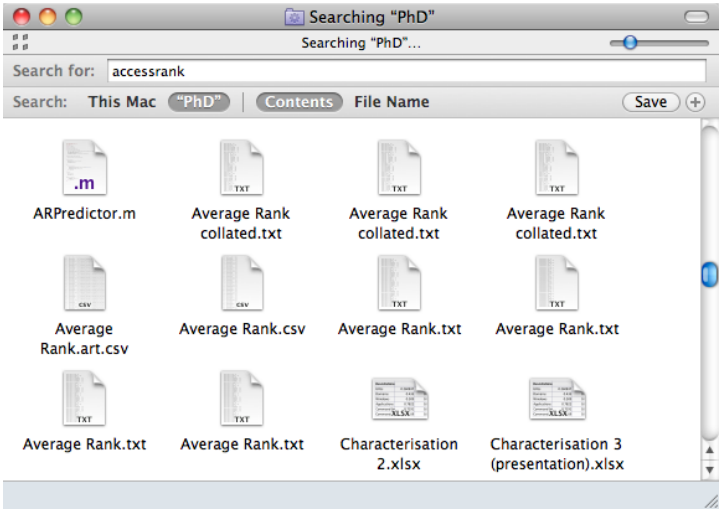
(a) Searching in the Start menu on Windows 7



(b) Searching in Windows Explorer on Windows 7



(c) Spotlight menu on Mac OS X 10.6



(d) Searching in the Finder on Mac OS X 10.6

Figure 2.5: Search interfaces

2.4.1 Saved Searches

Many modern file browsers, such as Windows and OS X, include the ability to save searches. The saved search (also known as *Smart Folders* on OS X) includes details about the search configuration, rather than the set of results, meaning that subsequent accesses produce up-to-date results that incorporate changes to files since the last time the search was used.

Windows 7 also includes support for *libraries*. Libraries are configured with multiple folder locations, and accessing them shows the content of all these locations together. While not as flexible as saved searches, libraries provide another means of providing dynamic content without the constraints of a strict tree structure.

2.4.2 Faceted Search

Search involves substantially different psychological processes than navigation. While navigation is predominantly recognition-driven, search is primarily recall-driven as it simply presents a subset of data in response to a search query.

Faceted search [114, 151, 91] aims to bridge this gap, allowing users to progressively filter a dataset by branching on attribute values. For example, starting at root level, a user could branch on files of type 'presentation', then branch again on files modified today, and then choose the target item from the resulting list. This approach improves flexibility by providing multiple paths to each item. Additionally, it solves a limitation of search by allowing users to explore a dataset; Koren et al. [114] state that *"While conventional keyword search queries no doubt have their place, they are not very conducive to exploration, since they require the user to possess some knowledge about the contents of the file system and relevant files in order to issue effective queries"*.

While faceted search is common on the web (e.g., [151, 90, 187]), it has not been adopted in any mainstream file retrieval tools. However, several researchers have created specialised file systems designed for faceted search. LISFS, or Logical Information System File System [149], uses logical formulas to filter data based on both intrinsic and extrinsic properties of objects. Similarly, Gifford et al. [79] describe a semantic file system that uses *virtual directories* to filter by attribute values. Intrinsic properties are computed from objects' content, while extrinsic properties are provided by the user. XMLFS [14] is a file system specifically designed for XML files, which branches on XML attributes in any order.

2.4.3 Research Systems

Many search-based systems have been proposed as alternatives to hierarchical file systems, typically to address deficiencies such as requirements to maintain a folder hierarchy and to store each file in a unique location. Several of these systems are heavily based on time attributes, demoting the importance of location. For example, Lifestreams [72] uses a timeline as its underlying metaphor. 'Streams' of files, ordered by time, can be searched to create smaller substreams. Lifestreams is based on a number of principles and observations, including that filenames and locations should not be required, documents should belong in as many places as seems reasonable, computers should make 'reminding' convenient and that systems should be able to summarise large groups of related documents in a concise overview.

MyLifeBits [77] is also heavily based on time attributes and is intended as a replacement for a file hierarchy. It focuses heavily on multimedia files and support for rich annotations. It uses a range of visualisations, particularly those based on time.

In TimeSpace [115], users create a set of activities. A primary view shows an overview of these activities, with recent files visible within each. An activity view shows all the files in a particular activity, ordered by time in the x-axis, and by type in the y-axis. Users have some flexibility to move items slightly in order to group related items together.

Placeless Documents [57] focuses on user-derived rather than system-derived properties. It assumes documents will be used by many users, and allows people to add personal properties specific to them. It also distinguishes between organisation for the purposes of grouping related files and document management for other purposes such as backup and sharing, noting that existing hierarchy systems assume one hierarchy can suit both purposes. Documents can have active properties containing code to achieve these goals, for example to backup, summarise the document after changes, or to log document accesses. Presto [58] is a prototype implementation of part of the functionality of Placeless Documents. It uses combinations of queries, inclusion lists and exclusion lists to return document collections.

Stuff I've Seen [61] is a search system focused on retrieving previously

accessed documents. It gives particular emphasis to filtering and sorting by the date documents were modified, and integrates files, email messages and web pages into its results. A precursor study to the Stuff I've Seen system found that the inclusion of landmarks on a timeline reduced retrieval times compared to a normal timeline that only provides dates [157].

Phlat [54] acknowledges the advantages of recognition over recall-based interfaces, and attempts to combine some of the benefits of navigation and search. After providing an initial text query, a range of filters can be used to iteratively filter results. File management events can be performed within search results – a feature lacking in most search interfaces.

Timescape [156] aims to replace the traditional file hierarchy by taking incremental snapshots of the file system. Users can specify a particular time, and they are presented with a view of what the desktop looked like at that time. The design encourages users to delete information that is not related to a current task, but allows users to retrieve deleted information by returning to an earlier snapshot. Users can also travel to a future time, for example to set reminders. While Timescape is designed as a document organisation system, it has many similarities to OS X's Time Machine backup feature.

While these systems are potentially beneficial, most require fundamental, potentially disruptive changes in users' workflows [121], and they have not been shown to be superior to existing hierarchy-based systems [35]. Nevertheless, they provide interesting insights into potential future directions for file retrieval interfaces.

2.4.4 Launchers

A number of third party *launcher* tools include functionality to type a query or command in order to quickly retrieve files (e.g., [31, 83, 100, 148, 161]). These tools – which often offer integration with a range of services, of which file retrieval is just one – are generally activated by a hotkey, after which users can type a few letters from a filename to launch the corresponding item or view a list of candidates. However, they are often restricted to certain types of files, such as applications, or data sources, such as recent items. As a result, they are optimised for certain subsets of files, and cannot usually be

used for all file retrievals.

Aceituno and Roussel [2] describe the Hotkey Palette, which allows users to assign hotkeys to particular files. The Hotkey Palette supports both global and contextual hotkeys. With contextual hotkeys, the hotkey is available only when a particular project is currently active, inferred by the active document's path. This feature allows the same hotkey to launch different files based on the current context.

2.5 Recommender Interfaces

Recommender interfaces automatically select a subset of files to present as candidate items to the user. The most common of these consider recency or frequency to determine their content. These interfaces are typically highly efficient for files that are contained within the subset, but of no use for files that are not. They work best when users can accurately predict whether an item will be in the subset; there is a high cost when these user predictions are incorrect, as illustrated in Figure 2.6.

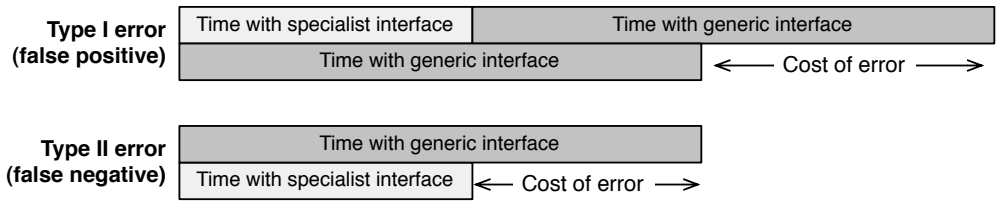
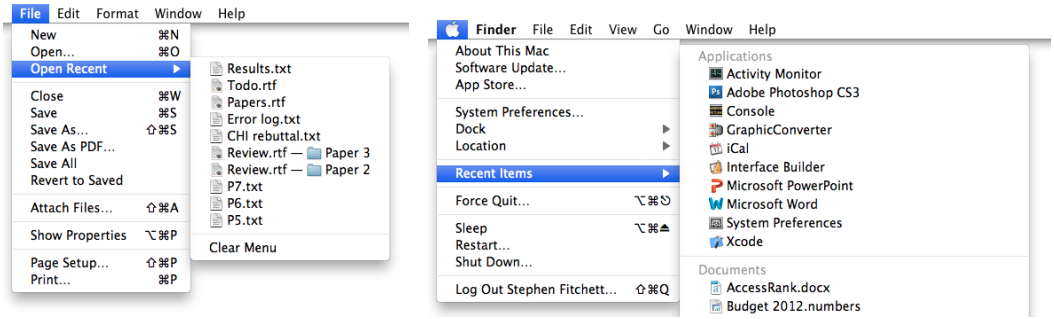


Figure 2.6: Cost of errors when attempting to use a recommender interface when the target item is not present (type I error for the hypothesis that use of the interface will be successful) and when not using it when it is present (type II error). Comparative task times are illustrative only.

Recommender interfaces differ from each other in *which* items they present, as well as *how* they present them – for example, how they are sorted or organised. The choice of both the selection algorithm and presentation method are important in reducing the probability of errors and allowing users to quickly determine if a target item is present.

The most common form of recommender interfaces are *Open Recent* menus (e.g., Figure 2.7a). Though they are sometimes presented in different forms



(a) Application's *Open Recent* menu (b) *Recent Items* menu in OS X's Apple menu

(c) Document Gallery in Microsoft Office for Mac 2011, which shows recent documents (d) Start menu in Windows 7

Figure 2.7: Interfaces for recent items

– the Word Document Gallery (Figure 2.7c) providing one example – they all allow quick access to recent items and are present in most document-based applications. While recency may not be the most accurate predictor, it allows for a relatively high degree of predictability for users, since users will generally have a good memory of which files they have most recently interacted with.

OS X also provides a system-wide recent items menu (Figure 2.7b). It differs from the *Open Recent* menu in several ways: (1) it includes items opened in any application; (2) it includes separate lists for applications, documents, and servers, rather than documents only; and (3) it sorts items in alphabetical order, rather than by recency as in *Open Recent* menus. This

last difference is perhaps motivated by the difference in context; with users often involved with multiple projects simultaneously, it may be difficult to remember the relative order of document accesses across multiple applications, whereas this task might be easier in a single application where there is a lower probability of interleaving tasks. In other words, the easiest attribute for scanning the lists for a target item likely differs based on context, with recency easiest within a single application, and filename in a global context.

Windows 7 also includes a global recent items list in the Start menu (Figure 2.7d, submenu visible on right), although it is not available by default and must be specifically enabled. The Start menu also includes a list of suggested applications on its left side, determined by an algorithm that incorporates both recency and frequency. This difference in selection method may again be inspired by context. Selecting items based on a combination of recency and frequency is likely to improve the probability that a target application is present in comparison to a pure recency algorithm, however this greater prediction accuracy might come at a cost of lower predictability for users – the use of a more complex algorithm means that users cannot easily guess whether a particular item will be included. Normally, this reduction in predictability would have a considerable cost (see Figure 2.6). However, the suggested applications appear in the same context that applications may ordinarily be accessed – accompanied by an "All Programs" submenu and search field – greatly reducing the cost of an error. With this lower error cost, the increase in prediction accuracy may outweigh the loss in predictability.

As another recommender interface, OS X provides a special "All My Files" view in Finder (Figure 2.8). By default, it categorises all the user's documents by type, sorting them by recency within each type. This effectively acts as a set of system-wide recent items lists for each document type.

2.6 Bookmarks

Many file retrieval tools include features that allow users to manually specify files that they want easy access to. These require effort to maintain, but potentially provide a more stable and relevant set of items than automatically generated sets.

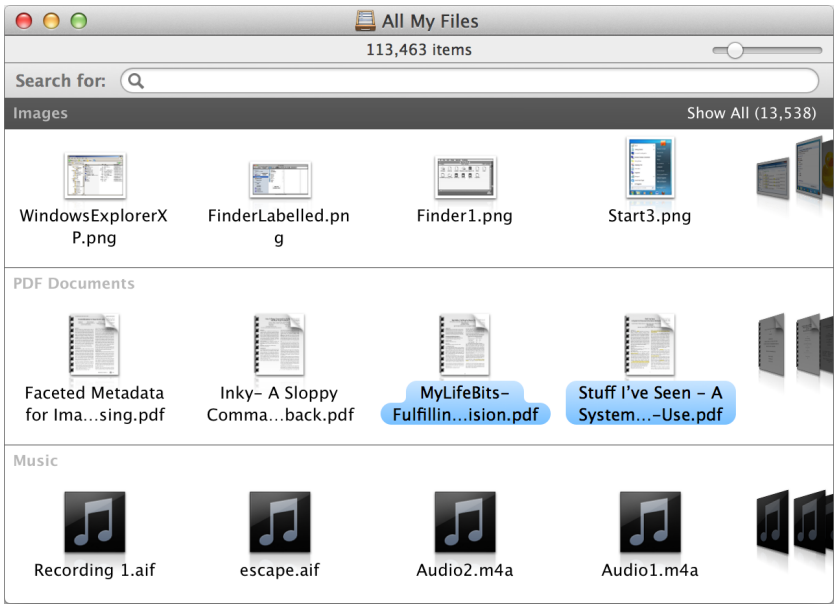


Figure 2.8: The 'All My Files' view in OS X 10.8

As an example, the OS X Finder provides several bookmarking features, two of which are shown in Figure 2.9. Folders can be dragged to or from a sidebar on the left of file browser windows. Less commonly, both files and folders can be added to the window's toolbar. However, the toolbar is primarily used for commands, and toolbar item names (such as folder or file names) are not shown by default. Folder bookmarking is supported in a similar way in Windows 7 (Figure 2.5b), with support for including folders in a 'Favorites' section in the sidebar.

Other bookmarking interfaces are designed specifically for applications. These include the Quick Launch toolbar (Windows XP, Figure 2.10a) and its successor, the Superbar (Windows 7), the Start menu (Windows 95 to Windows 7) and the OS X Dock (Figure 2.10b). The Superbar and Dock also double as application switchers, and provide indicators as to which applications are running.

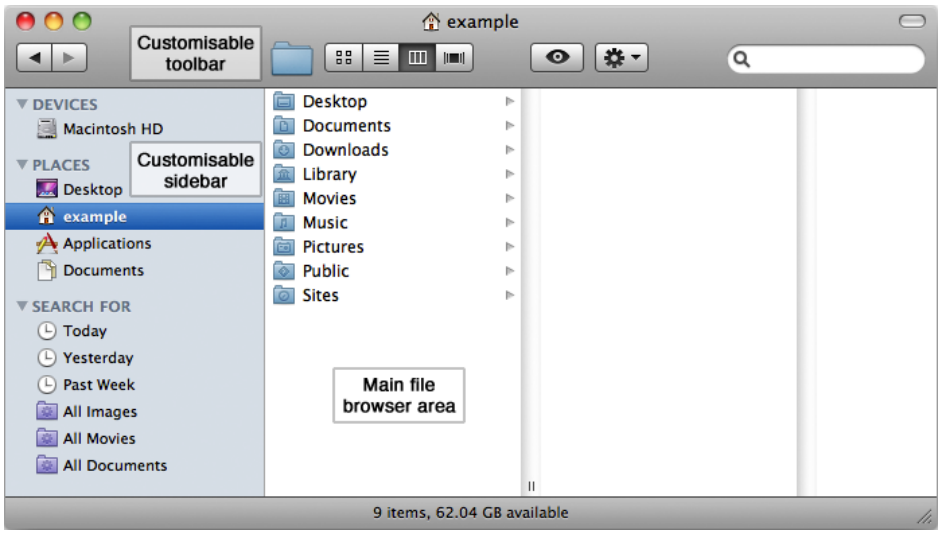


Figure 2.9: Bookmarking features in the Finder in Mac OS X 10.6

(a) Quick Launch toolbar in Windows XP

(b) Mac OS X 10.6 Dock

Figure 2.10: Customisable application launching interfaces

2.7 Conclusion

This chapter provided an overview of existing file retrieval methods. This included a review of navigation-based methods, interfaces that recommend items (such as 'Open Recent' menus), interfaces that provide fixed access to a set of custom items, and a variety of search interfaces. As well as traditional search systems that provide a list of results in response to a query, the chapter described alternatives that provide greater integration with navigation-based methods, such as saved searches and faceted search, research systems that aim to replace the hierarchical paradigm, and third party launcher tools such as those activated by a hotkey. File retrieval is further explored in Chapter 3, which examines the broader topic of organisation and retrieval behaviour.

Chapter III

An Overview of Organisation and Retrieval Behaviour

Chapter 2 described the range of file retrieval methods available to users, with the focus on the *tools* rather than the *users*. Conversely, this chapter focuses on the users: how do they organise and retrieve their files, and how do organisation and retrieval strategies for files compare to other domains such as email management and webpage retrieval? Section 3.1 begins by comparing the properties of common domains in Personal Information Management: personal files, the web, and emails. Sections 3.2 and 3.3 then describe web and email management strategies in detail. Section 3.4 briefly reviews research on the management of paper documents – in many ways a precursor domain to electronic file management.

Section 3.5 then provides a detailed review of electronic file management. This review includes literature related to each of the file management subtasks described by Barreau [18]: acquisition of items, organisation of these items, maintenance of information, and information retrieval. The section discusses a range of topics, including: different types of information, each with different temporal properties; representations of structure, including differences between user and system conceptualisations of information and debate about whether files should be able to be stored in multiple locations; organisation and maintenance strategies; and retrieval behaviour, with reference to underlying psychological principles.

3.1 Comparison of Domains

The methods that people use to retrieve their files are strongly influenced by properties of the file management domain. These properties differ from those of similar domains, such as email management and webpage retrieval.

Property	Personal Files	Emails	Web
Exhaustive hierarchy	✓	✗	✗
Familiar hierarchy	✓	✓	✗
Specific target	✓	✓	✗
Interlinked	✗	✗	✓
Communal access	✗	✗	✓
Frequent revisitations	✓	✗	✓

Table 3.1: Properties that influence retrieval methods in different domains

Table 3.1 summarises properties of these domains that affect retrieval strategies.

Familiar and exhaustive hierarchies encourage use of navigation. Without them, users must resort to other methods (when hierarchies are not available) or to guessing a target's location in a hierarchy (if they are unfamiliar). For personal files, users are forced to place every new file in a specific location of their choice, so the hierarchy is both exhaustive (every file exists in the hierarchy structure) and familiar. New emails arrive in an inbox, and are not filed into a hierarchy unless the user specifically chooses to do so or has set up filters in advance; emails thus have a familiar hierarchy, but it is unlikely to be exhaustive due to the considerable effort required to create such hierarchies [185] relative to the benefits they offer [184]. Web content is fundamentally different; the content is maintained by others, and is thus largely unfamiliar. While some web directories are available, they are not exhaustive. Hierarchical navigation is therefore not feasible for web content except within limited contexts, and unfamiliar content encourages the use of search [25, 155].

Specific targets demand precise retrieval. Retrieval tasks for personal files and emails typically involve a specific file or email, such as a document to edit or an email that specifies the details of a meeting. It is not sufficient to find similar files as they will not satisfy the user's goal. On the web, however, users are often (though not always) looking for a particular piece of *information* rather than a particular web page – for example, a weather forecast or product advice. In these cases, many different webpages will contain this information, and the user will complete their goal by retrieving

any one of them. This provides a greater level of flexibility to retrieval tools.

Another property that aids retrieval with web content is that websites are often *interlinked*. This provides greater flexibility again; if no search results contain the desired information or content, one with related information might *link* to a page that does. Websites also have *communal access* – unlike personal files and emails, large numbers of people are accessing the same items. While this does not *directly* affect retrieval, it allows for optimisation of search and prediction algorithms based on aggregate patterns of retrieval, providing another advantage to web search features. Neither personal files or emails have these properties.

Finally, *revisitation frequency* affects the ease in which items can be found. If an item is frequently retrieved, people learn the physical mechanisms required to retrieve it, and improve in efficiency [117]. These mechanisms could include hierarchy locations, search terms, web URLs, screen locations or other information. Revisitation is common for personal files, which are often edited over time. Web pages are often revisited to see new content [3], or increasingly, to interact with other users. Revisitation rates for web pages have been found to be between 58% and 81% [174, 47]. On the other hand, revisitation is less common for email; Elsweiler et al. [64] found that only 6-15% of interaction chains involved revisiting an email, and that the largest cluster of users revisited an email only once every four days.

These important differences mean that the choices that users make when selecting an appropriate retrieval method are fundamentally different depending on the domain. Organisational strategies are also dependent on anticipated future retrieval behaviour [185]. Accordingly, properties that influence retrieval behaviour also affect how users organise their content in each domain – for example, when revisitation is rare, users will spend less effort creating a comprehensive hierarchy structure. Boardman and Sasse [34] found large differences between the hierarchy structures in different domains. Only 3% of files were not filed into hierarchy locations (instead being left in some default location, such as the desktop), compared to 39% of bookmarks and 42% of email messages. Similarly, the average depth of files in their hierarchy was 3.3, compared to only 1.7 for email messages and 1.3 for bookmarks. Users were most likely to organise their files, followed by emails

and then bookmarks; none of Boardman and Sasse's participants organised emails who did not also organise files, nor did any organise bookmarks who did not also organise files and emails.

Several researchers have examined what Bergman et al. call the *Project Fragmentation Problem* [23, 104, 155] – that is, the forced maintenance of multiple hierarchies based on format (i.e., for files, emails and bookmarks) when there is content corresponding to particular projects distributed across them. Bergman et al. [23] noted that users were more likely to refer to their content in relation to a project, rather than a format. Furthermore, they found that 20% of folders had another folder relating to the same project in a different hierarchy. Similarly, Boardman et al. [35] found a 21% folder overlap between files and emails, and Boardman and Sasse [34] found an average overlap of 7.0 folders between file and email hierarchies, compared to 2.9 between file and bookmark hierarchies and 2.8 between email and bookmark hierarchies. An average of 40% of these overlapping folders were based on projects (e.g., 'PhD') with 27% based on role (e.g., 'Teaching'). Boardman [33] speculates that these overlaps would be even larger if users did not have the overhead of managing parallel hierarchies. Accordingly, several approaches have been suggested that use a single hierarchy [23, 33, 35]. It remains to be seen whether they would be effective in practice, however, in light of the significant differences between properties of the domains.

This research on hierarchy fragmentation shows that there is considerable overlap between hierarchies from different domains. An understanding of retrieval behaviour in each domain can therefore provide useful insights that are relevant to file management strategies. To assist in this, the following sections detail web and email retrieval, followed by detailed discussion of file organisation and retrieval.

3.2 Refinding on the Web

Accessing websites is an important activity in Personal Information Management. These website accesses can be categorised as either *finding*, which involves exploratory activity to find relevant content that has not previously been accessed (involving recognition only), or *refinding*, which involves both

recognition and recall to access a previously visited website [39]. Refinding (another term for revisitation) is particularly common on the web; it has a 44-81% revisitation rate [174, 136, 147, 135], partly as a result of websites which offer changing content over time (such as news websites). In terms of relevancy to personal file retrieval, refinding is also of the most interest, as it is the subset of website accesses that – like personal file retrieval – deals with familiar content.

The web has evolved tremendously over just a couple of decades. In one of the first studies of website retrieval, conducted in 1994 before the advent of search engines, there were only 7300 web servers in existence, and users averaged just 14 page accesses per day [42]. This latter figure grew to 21 in 1996 [174], 41 in 2000 [136, 47], 90 in 2005 [147, 182], and is undoubtedly even higher today. It is therefore important that users are provided the tools they need to efficiently visit – and revisit – this increasing amount of content.

Web browsers and specialist websites over a range of features to assist users in revisiting web pages. Browsers include features such as bookmarks and history lists, as well as providing back and forward buttons to navigate through the most recently visited pages. These buttons are especially important considering that three quarters of revisits are to the ten most recently accessed pages [174] and 73% are for pages visited in the last hour [147]; indeed, the back button is used for 31% of page revisits [147]. Web browsers also include autocompletion features when typing in their address fields, as well as often providing a list of suggested locations. Finally, users can refind previously accessed pages, as well as discover new pages, using search engines.

Bookmark collections – the area of web refinding most analogous to organisation of personal files – are of particular interest. Average bookmark collections have been shown to contain close to 200 bookmarks [136], with new bookmarks added every 4-5 days, on average [136, 1], but bookmarks deleted only an average of once every 32 days [136].

Abrams et al. [1] analysed the way users organise their bookmarks. They found a strong relationship between organisational tendencies and the size of bookmark collections. Few users with under 35 bookmarks organised their collections. Those with 101-300 bookmarks were most likely to have a sin-

gle level of folders, while 44% of those with over 300 bookmarks had collections with multiple hierarchy levels. Roughly half of all users organised their bookmarks sporadically, with 26% never organising them and 23% filing bookmarks as soon as they were created; this latter figure rose to 67% of users with over 300 bookmarks. Bookmarks were often also organised for quick retrieval of the most useful bookmarks, for example by placing them at shallower locations in the hierarchy. Users created bookmarks for several reasons – most commonly for archival purposes, but also to create short-term reminders or shortcuts to frequent locations. Boardman and Sasse [34] found that users employed multiple bookmark management strategies, often based on the purpose for the bookmark. For example, bookmarks created as short-term reminders were less likely to be filed.

Users face substantial problems maintaining bookmark collections [105]. Their contents frequently becomes outdated, both because of changing information needs [1] and changes in the underlying data that bookmarks link to – a quarter of all bookmarks are no longer valid, and 5% are duplicates [136]. Further complicating maintenance, default bookmark titles are often poor descriptors of their content, yet few users go to the effort of changing them [1]. Users also face challenges maintaining consistency between hierarchies of bookmarks and other content, such as files and emails [107].

Perhaps as a result of these challenges, bookmarks are rarely used to revisit websites [35, 105]. The most optimistic rate of use is 18% of web revisitations, achieved in a lab study [106]; field studies have provided much lower figures, in the low single digits [42, 147, 174]. One reason for these low usage rates is that bookmarks provide no reminding function, limiting their utility for a common use case; some users instead opt for other approaches such as to send themselves the page URL in an email [105]. On the other hand, Abrams et al. reported that 96% of bookmarks had been accessed in the last year [1].

Obendorf et al. [147] found that direct access methods, including bookmarks, history tools and autocompleted URLs, are the most common browser features used to revisit pages last accessed between an hour and a day ago, while the back button is most common for short-term revisitations (those visited in the last hour) and hyperlinks are most common for long-term re-

visitations (last visited over a week ago). Of these methods, history tools – which provide a list of previous page accesses, ordered by recency – are the least used, with several researchers finding that users rarely use them in practice [105, 38, 174].

External to the functionality offered by a particular web browser, search engines provide powerful functionality for discovering new content, but are often commonly used for revisitation [13]. Capra and Pérez-Quiñones [39] found no significant difference between frequency of use of search engines for finding compared to refinding, and Teevan et al. [175] found that up to 40% of search engine queries were conducted for the purposes of refinding and nearly 30% of URLs clicked in search results were clicked multiple times by the same user. Furthermore, 24% of queries were *navigational queries*, defined as queries where a single result is selected and where both the search query and result selection are identical to that of an earlier query. Repeat page accesses were less likely, however, when item ranks changed, indicating the importance of stability in search results. In addition to these navigational queries, Tyler and Teevan [180] found that many refinding queries are often shorter than their initial finding queries and rank the target item higher, suggesting that people learn information about the pages they visit that helps them when they later search for them again.

3.3 Email Management

Email management is a topic of considerable importance, with users dealing with increasing amounts of incoming email that consumes a significant portion of their time to process. The size of email archives increased ten-fold between 1996 and 2006 [185, 69], with many users receiving close to a hundred emails each day [69]. However, there is considerable variation in management strategies, with low revisitation rates making it difficult to justify complex organisational schemes for many users; Elsweiler et al. [64] found that only 3.6% of messages were ever revisited. Additionally, users often find it difficult to maintain a suitable folder structure due to the difficulty in imagining future retrieval requirements [185], with 35% of email folders containing fewer than three messages, and duplicate folders also common [185]. Another bar-

rier to organisation is that tasks often involve the participation of others and cannot be actioned until a response is obtained [21].

Several researchers have investigated common strategies for email organisation. Mackay [124] identified two strategies, which are not mutually exclusive: *prioritisers*, who use a set of rules (either manual or automatic) to sort email messages based on priority, and *archivers*, who maintain a large number of folders that are subject-based, rather than priority-based. Whittaker and Sidner [185] classified email users as either *no filers* (who do not sort emails into folders, instead relying on opportunistic retrieval methods such as search), *frequent filers* (who minimise the size of their inbox by frequently filing messages into folders) and *spring cleaners* (who periodically sort their inbox into folders). Elsweiler et al. [64] found users were split roughly evenly between these strategies. Fisher et al. [69], however, found that although there was substantial variation between users in terms of use of folders and inbox size, there was no clearly discriminable groups, with most users clustered in the middle. Similarly, Boardman and Sasse [34] found that most users employed multiple management strategies and could not easily be categorised using the strategies identified by Whittaker and Sidner.

While many users make substantial use of folders to sort email [185, 69], the effort required to maintain this organisation is significant [16], with the time spent organising email messages more than the time spent later retrieving them [21]. In fact, despite the organisational effort required, organising emails into folders provides little benefit; Whittaker et al. [184] found that those who frequently file emails are no more likely to revisit emails or successfully find them when they do, and are slower to revisit items overall than those who do not file extensively. Indeed, it is common for users to spend significant time filing messages which are never viewed again [34]. Bälter [16] found that using no folders is the most efficient strategy for many users, particularly those with over a thousand stored messages. Despite this, most email revisitation involves the use of folders (often in combination with search) [64]. Regardless of the amount of email organisation involved, opportunistic retrieval behaviours (such as search) dominate [184], implying that the benefits provided by the maintenance of folders are minimal.

Elsweiler et al. [64] explored sequences of events that formed *refinding*

chains, the purpose of which was to revisit previously read emails. They found that just 36% of revisitations were for emails more than an hour old. Users often became 'lost', viewing single messages multiple times in two-thirds of refinding chains, and single folders multiple times in 30% of chains that made use of folders.

Use of search is more common in email than for files [34], with high success rates [184] and use in over 90% of refinding chains, though it is largely used to narrow down the search space rather than to immediately find a result [64]: indeed, many emails are viewed for each query as part of an orienteering strategy used subsequent to search [64, 176], contrasting with results for web searches where most attempts to search for the purposes of revisitation involve the selection of a single result [180]. This behaviour perhaps explains why advanced search is rarely used to retrieve emails [64], as users prefer to browse through a narrowed-down search space (for example, filtered by sender) rather than set up elaborate queries. Email, where a combination of search and orienteering are often used, therefore sits in a middle ground between file retrieval, where navigation is preferred [25], and the web, where search is preferred [106].

Though the low benefits and high costs of email organisation, combined with its adoption by only a subset of users, are partly an artefact of the retrieval properties of emails (discussed earlier in this chapter), findings on email management still have implications for file retrieval. In particular, they demonstrate that when organisation is optional, it is not necessarily adopted, nor beneficial. These findings are similar to those that have found users do not like to manually annotate their files [52], and adds support for those who argue that users should not be forced to name and file documents on creation [72]. These topics are discussed in greater detail later in this chapter.

3.4 Management of Paper Documents

Many researchers have examined how office workers manage paper documents (e.g., [118, 41, 125, 183, 127]). While the domain of paper document management does not offer the same organisational and retrieval features as are possible in an electronic setting, it does share many properties with its

electronic equivalent. In particular, the only significant difference between the two domains, in terms of the properties listed in Table 3.1, is that paper documents are freed of the strict requirement so be sorted into a file hierarchy. Instead, they can be stored loosely, with spatial location serving a much greater role than exists in an electronic environment. Paper documents can be organised in piles [121], with the order of the items in the pile potentially serving an informative role, whereas items in an electronic folder do not generally possess an inherent order property and can normally only be sorted by the value of some attribute [108]. Additionally, electronic files can be stored in a hierarchy of arbitrary depth, while it is rare (due to physical constraints) for paper folders to contain further folders.

Despite these organisational differences, there are nevertheless substantial similarities between the two domains – in particular, because the *content* is similar across domains, with the primary difference being only its *form*. As an example of one similarity, Malone [127] found that an important purpose of desk organisation is to remind about current tasks, similar to the way in which users use computer desktops (discussed later in this chapter). Paper document management strategies also help inform how people manage their documents when unconstrained by certain limits imposed by an electronic environment.

Whittaker and Hirschberg [183] conducted a study of the paper document collections of workers who were shifting offices. They found that 49% of documents were unique, while 36% were copies of publicly available documents. These latter documents were kept for several reasons: so that they were readily available; as reminders; due to a lack of trust that they would remain in external stores (for example, they could be removed from a website); and sentiment. The remaining 15% of documents were unread.

Though participants were aware that the value of particular information decreased over time as circumstances changed, they ended up with large archives deliberately, and not because of a lack of time to sort through their information. Archives were rarely cleaned up spontaneously, with 84% resulting from external events such as job changes or office moves. During these clean-ups, almost a quarter of discarded items remained unread; Whittaker and Hirschberg concluded that people often engaged in deferred evaluation,

resulting in large amounts of data that is not discovered to be superfluous until circumstances such as an office move require them to return to it. This was in part an exercise of risk management: though people often perceive information as being of low value, they retain it just in case it later turns out to be useful.

This issue is likely to be exacerbated in the domain of electronic files, for several reasons. First, information can be filed away at arbitrarily large depths in the hierarchy, making it easy to file and forget. Second, the information takes up no physical space, making large collections of files less obvious, particularly with ever-expanding disk capacities. Third, electronic storage increases the portability of information, meaning that there are potentially fewer extrinsic events that necessitate the kind of clean-up that Whittaker and Hirschberg describe. As a result, electronic file collections have the potential to grow increasingly large, creating significant challenges for organisation and retrieval of electronic documents. These challenges are described over the remainder of this chapter.

3.5 Electronic File Management

The dominant file organisation system in modern computer systems is that of a file hierarchy – files and folders nested inside other folders. Some systems and devices, in particular mobile devices, use more constrained organisation schemes, such as those that maintain separate sandboxed areas for each application. Other approaches that try to replace file hierarchies, such as faceted search, are described in Chapter 2. However, the primary focus of this thesis is on retrieval in file hierarchies, as these remain the predominant system for managing large repositories of personal files.

File collections on personal computers are maintained primarily by a single person, unlike on the internet where individual users have little or no control over structure. This difference has a large effect on how people organise their files, as people consciously organise their files for easy retrieval [19]. In particular, users give careful attention to file naming, but primarily for the purpose of jogging their memory during recognition rather than for recalling the name when searching [18, 145].

This section begins with a summary of previous studies of file management. Section 3.5.2 discusses the types of information that users work with, and how organisational and retrieval behaviour differs for each. Section 3.5.3 discusses the structure of personal file collections, including a description of the conventional hierarchical structure, debate about support for storing files in multiple locations, and how system models differ from users' conceptual models of their information. This is followed by Section 3.5.4, which describes different strategies for organisation and maintenance of file collections. Section 3.5.5 details how users make use of the desktop and other locations that allow (or can be set to allow) spatially consistent item locations. Finally, Sections 3.5.6 and 3.5.7 discuss the memorability of different file properties and users' retrieval preferences and behaviour.

3.5.1 Summary of File Management Studies

There have been a considerable number of studies on file management behaviour. Table 3.2 summarises 28 previous studies, outlining the aspect of file management they investigated (e.g. organisation or retrieval), the data collection techniques used (e.g. interviews, system snapshots or lab studies), the platforms or mediums under investigation, and the sample size. The majority of these studies focused of file organisation, while just 10 included analysis of retrieval behaviour. Many of these studies had retrieval as only a partial focus (e.g., [145, 155]) or focused on particular aspects of retrieval (e.g., [26]).

Owing to the methodological difficulties in evaluating and characterising use of personal information management tools [104], the predominant data collection methods used in these studies were interviews and surveys, used in 23 of the 28 studies. Screenshots and system snapshots were used in eight of the organisation studies, although this technique was not amenable to studying retrieval behaviour. Instead, studies of retrieval often relied on lab studies or use of controlled retrieval tasks (five out of ten studies). Four studies used observation (for analysis of both organisation and retrieval), and just one study made use of a logging tool [102].

Study	Aspect	Data Source	Medium	Sample
Barreau (1995) [18]	Organisation & retrieval	Interview & observation	Electronic (various)	7
Bergman et al. (2006) [23]	Organisation (fragmentation)	Interview & screenshots	Electronic (Win/Mac)	20
Bergman et al. (2008) [25]	Retrieval	Questionnaire	Electronic (Win/Mac)	78/47/589
Bergman et al. (2011, 2012) [29, 30]	Retrieval	Experiment	Electronic (Win/Mac/Linux)	296
Bergman et al. (2013) [26]	Organisation & retrieval (folders vs. tags)	Interview, snapshot & controlled tasks	Electronic (Win)	23
Bergman et al. (2013) [27]	Retrieval	Lab study	Electronic (Win)	62
Blanc-Brude & Scapin (2007) [32]	Retrieval	Interview & lab study	Electronic & paper	14
Boardman (2001) [33]	Organisation (fragmentation)	Interview	Electronic (Win/Mac/Linux)	10
Boardman & Sasse (2004) [34]	Mixed (fragmentation)	Interview & snapshot	Electronic (Win/Mac/Linux)	31/8
Boardman et al. (2003) [35]	Mixed (fragmentation)	Interview	Electronic (Win/Mac/Linux)	25
Case (1986) [41]	Organisation	Interview	Paper	36
Golemati et al. (2007) [80]	Retrieval	Interview & lab study	Electronic (Win)	18/15
Gonçalves (2002) [81]	Organisation	Questionnaire	Electronic	88
Gonçalves & Jorge (2003) [82]	Organisation	Snapshot	Electronic (Win/UNIX)	11
Henderson (2005) [92]	Organisation	Interview & Snapshot	Electronic	6
Henderson (2009) [94]	Organisation	Interview	Electronic	115
Henderson & Srinivasan (2009) [95]	Organisation	Snapshot	Electronic (Win)	73
Jensen et al. (2003) [102]	Organisation (provenance)	Survey, logging & observation	Electronic (Win)	24
Jones et al. (2005) [108]	Organisation	Interview & snapshot	Electronic	14
Kaptelinin (1996) [112]	Organisation	Interview	Electronic (Mac)	12
Kwasnik (1989, 1991) [118, 119]	Organisation	Interview & observation	Paper	8
Malone (1983) [127]	Organisation	Interview	Paper	10
Mander et al. (1992) [128]	Organisation	Interview	Paper	13
Nardi et al. (1995) [145]	Organisation & retrieval	Interview	Electronic (Mac)	15
Ravasio et al. (2004) [155]	Organisation & retrieval	Interview	Electronic (Win/Mac)	16
Tang et al. (2007) [173]	Organisation (commonality)	Snapshot	Electronic (Win/Linux)	15
Teevan et al. (2004) [176]	Retrieval	Interview & observation	Electronic	15
Whittaker & Hirschberg (2001) [183]	Organisation	Survey	Paper	50

Table 3.2: Summary of studies on file management

3.5.2 Types of Information

The ways in which people interact with their files depends in part on the role of the information contained within them. For example, Cole [49] noted the following types of information in a study of paper documents, while Nardi et al. [145] observed the same types in their study of file organisation:

Ephemeral information is temporary information such as "to do" lists, memos and news articles. In their study, Nardi et al. found that users tended to keep this information either immediately visible or very loosely filed, for example by placing files on the desktop. Often information would be created as ephemeral information but would persist for longer than expected and clutter the user's filing system. This may be because the cognitive difficulty of deciding how to classify information acts as a barrier to filing it [127], resulting in more information being classed as ephemeral than perhaps should.

Working information is frequently used information related to current work needs, with a shelf life of weeks or months. It is typically organised fairly well in the file hierarchy. Users have little difficulty finding such information as it is frequently accessed and they can therefore easily remember its location.

Archived information is infrequently accessed and has a shelf life of months or years. It would typically represent completed work. Barreau and Nardi [19] argue that it is often not filed well, since the effort to create elaborate filing schemes is deemed higher than the information is worth. Boardman and Sasse [34] also found that items were rarely archived, but found that the information *did* have worth – albeit erratically. Extensive hierarchy maintenance was most often conducted during significant life changes, such as starting a new job. Fertig et al. [66] also contend that situations occur when old information is essential and Cook [50] argues for the importance of archiving information in organisational settings.

Boardman and Sasse [34] argue that these classifications are misleading – particularly *archived information*, as many users do not explicitly archive. They propose an alternate classification based on two dimensions:

Information usefulness – classified as *active* (including both ephemeral and working information), *dormant* (inactive information that is potentially useful), *not useful* and *un-assessed* (new items that have not yet been inspected).

Information ownership – classified as *mine* or *not-mine*. The former includes not just user-created items, but also those to which the user has attached value, such as filed emails. The latter includes those items which the user does not have direct ties to – for example, information on the internet. This classification has a strong relationship with the *familiarity* property of the domain classification earlier in this chapter.

As Nardi et al. [145] found, these different types of information often require different organisational structures. However, this creates a tension between organisation for current use and for later reuse, as files often move between these categories [108]. Files currently under frequent active use should ideally be easy to retrieve (e.g., at a shallow hierarchy location), while archived files that are likely to be accessed only infrequently may require better organisation in order to assist users in finding them. These uses imply different organisation structures, which can result in complex hierarchies that require an increased level of maintenance as information needs change.

Files classified as archived information (Nardi et al.'s classification) or as dormant or not useful (Boardman and Sasse's classification) are often the most difficult to retrieve, since users are less likely to remember their exact locations or filenames. Navigating to these files is troublesome as a lot of trial and error is involved, while search is even more difficult since users must remember a file's name or properties to search for it. In his study of physical organisational systems, Malone [127] probed study participants to find documents with descriptions given by coworkers. In two thirds of these probes, the documents were not filed under the dimensions (such as title, author or person it was about) used to describe them. Furthermore,

he observed that the cognitive difficulty of classifying information acted as a barrier to filing.

3.5.3 Representation of Structure

File hierarchies are the predominant way to organise files on modern computer systems. Though they face several limitations – discussed throughout this section – users have generally favourable opinions of them. For example, Jones et al. [108] asked participants whether they would prefer a hierarchical system with folders, or a flat system where files could be accessed using a search-based mechanism, with only one participant responding positively to the potential removal of folders. Similarly, Bergman et al. [26] showed a preference towards hierarchical storage for both files and emails, and found that only 4% of files were stored in default locations (e.g., "My Documents" on Windows).

Several studies have examined how people structure their file hierarchies. In general, hierarchies are broad, shallow, and often unbalanced [158, 33]. Gonçalves and Jorge [82] analysed the structure of file hierarchies of 11 participants, examining only portions containing user documents. Users averaged about 8000 files within these, however there was considerable variation. They found folders contained an average of 13 files, had a branching factor (the average number of subfolders at a given tree level) of 1.84, and that the hierarchies were fairly well balanced. The hierarchies had an average depth of 8.45. In an analysis of filenames, they found that 60% of filenames contained numbers, but only 0.33% contained dates. Filename lengths averaged 12.6 characters, however they are likely longer in modern systems as file systems no longer impose tight constraints on filename lengths.

Henderson and Srinivasan [95] ran a more recent and larger scale study of Windows XP users, again analysing portions of hierarchies that contained user documents. They found similar results to Gonçalves and Jorge: 5850 documents per user, an average tree depth of 9.65, folders containing an average of 11.1 files and a branching factor of 1.93. They also found that 74% of folders did not contain any subfolders, but the folders that did averaged 4.1 subfolders each. 7.9% of folders were completely empty. When performing

name comparisons, 21.8% of filenames were duplicates, as well as 23.5% of folder names. Although the average maximum tree depth was 9.65, average depths within the trees were a considerably smaller 3.4.

The Role of Folders

While an important function of folders is to organise files with the goal of aiding later retrieval, they have additional uses. Jones et al. [108] found that folders provide information in their own right, such as the decomposition of a project, and that facilitating revisitation is not necessarily the primary purpose of folder organisation. They also identified several limitations of folder hierarchies. For example, while folders can be sorted within a view based on common properties such as name or modification date, they do not typically support any manual ordering, and users often include leading characters in folder names to force a particular order.

The folder metaphor is based on a physical folder, containing a set of paper documents and typically with an assigned name. However, paper documents can also be grouped into more informal *piles* – a stack of documents with no attached name. Piles can be an effective organisational tool for paper documents [127, 121], however they are not commonplace in electronic systems. Lansdale [121] expressly dismisses the idea of electronic piles, stating that the use of piles is used as a "compensating strategy for the problems of classification" and implying that the benefits offered by computing environments negate any benefits that piles offer in a paper setting. Nevertheless, Malone [127] advocates the use of piles in an electronic setting as a way to ease the cognitive burden of classifying information. This view is supported by Mander et al. [128], who found that users are sometimes dissatisfied with the requirement to make explicit categorisation decisions. They explored possible support for piles in an electronic setting, and argued that piles are more suitable than hierarchical folders for items that do not require detailed categorisation, as might be the case for some ephemeral or working information. They found that the ability to reorder items in a pile provided significant utility; for example, users could order items based on priority. Another benefit of paper piles is that they provide an indicator of

their contents just by looking at them. This benefit is implemented in OS X's *Stacks* feature, which replaces the icons of folders in its Dock with those of their contained files, drawn in a stack formation. However, these stacks retain the traditional requirements of folders, such as requiring an explicit name. A more complete piling metaphor was implemented in BumpTops [4], a pen-based desktop designed to more closely resemble the physical desktop.

User Versus System Representations

Users can most easily recall file locations and retrieve files when their conceptual model of their files is consistent with the system's representation. By organising their files themselves, they are able to approximate some level of consistency. However, there remain differences between the way users think of information and how systems represent it [155, 22]. In particular, several aspects of system design create barriers to full compatibility between the two representations:

Structural hints – Systems typically create an initial hierarchy, hinting to a user how they should organise their files. For example, both Windows and OS X provide separate folders for a user's documents, pictures, and movies, suggesting that users organise based on file type, whereas organisation based on type is normally quite rare [92] and users are more likely to think of their content in terms of project or role rather than format [23, 118]. While these initial hierarchies do not necessarily force a particular structure, they strongly suggest it, making it less likely that the eventual hierarchy will match the user's conceptual model and impacting their sense of ownership over the hierarchy structure [93, p. 233].

Forced dimensions of organisation – Systems often impose constraints on organisation in order for certain behaviour to occur. For example, dedicated folders may be shared with other users or backed up [57]. Again, this forces users to organise by dimensions that may be inconsistent with their conceptual models [23]; for example, they may wish to share only some files from a particular project, requiring them to split the project amongst multiple locations.

Forced model - The hierarchical file structure assumes that users think of their content in a strict hierarchical fashion. While users generally prefer file hierarchies to alternative representations [108, 26], it prevents or limits other representations, such as those where files are classified along multiple dimensions (further discussed in the following section). Lansdale [121] emphasises this point, stating that "information does not fall happily into neat categorisation structures which can then be implemented on a system by using simple labels".

Bergman et al. [22, 24] advocate the *User-Subjective Approach*, which suggests that Personal Information Management systems should be designed to better relate to subjective attributes. In particular, the approach includes three generic principles: *the subjective classification principle*, which states that items should be grouped by subjective topic rather than technological format; *the subjective importance principle*, which states that an item's subjective importance should determine its level of salience and accessibility; and the *subjective context principle*, which states that information should be retrieved and viewed in the same context as it was previously. In particular, the first principle suggests that the first two barriers outlined above impose unnecessary technological constraints, with a design that does not consider a user's conceptual representation of their files.

Multiple Storage Locations

There has been considerable debate amongst researchers about the requirement imposed by hierarchical-based systems of storing files in a single location. Though this is representative of the constraints that exist in a paper-based system, many researchers have noted limitations in this strategy [37, 72, 58, 114, 127, 153]. Referring to physical storage in 1945, Bush [37] stated that "[data] can be in only one place, unless duplicates are used; one has to have rules as to which path will locate it, and the rules are cumbersome", noting that the human mind does not work that way. More recently, Freeman and Gelernter [72] argued that breaking the desktop metaphor to allow for documents to be stored in multiple places would be beneficial. They stated that "paper can't be in more than one place, but electronic documents

can (or can appear to be)" and "documents should belong to as many [directories] as seems reasonable, or to none". Dourish et al. also commented on the limitation, stating that it limits files to a single spot in the semantic structure [58] and arguing that documents are often relevant to multiple roles and activities [57]. Koren et al. [114] argue that there are often multiple reasonable places to file a document, especially as their diversity increases. If a user is to come up with a different answer to the question "where should this file be located?" at saving and retrieval times, locating the file will be difficult. Quan et al. [153] demonstrated many of the potential benefits of a system that supports multiple classification, showing that a bookmark management system with such support resulted in both faster organisation and retrieval times.

While major operating systems do not provide a way to physically store files in multiple locations, they do provide mechanisms to work around this limitation [133]. Microsoft Windows offers shortcuts, UNIX offers symbolic links, and OS X offers both aliases and symbolic links. Fundamentally, while they have different names and implementations, they all achieve the same goal: they point by reference to the original item, allowing multiple paths to a target file or folder. This differs from filing items in different locations, however; for example, deleting a reference simply deletes the link to the original, while deleting the original file breaks all links to the file. While these help with developing more logical file hierarchies, they are rarely used in practice [82, 155], in part because they require effort to create and maintain [112]. Their most common use is to link to applications and folders from the desktop [112].

Bergman et al. [26] performed a comprehensive comparison of the use of folders (which require items to have a single location) and tags (which allow for multiple classification) in email and file domains. They found that few participants used tags, even after a two week period when they were forced to categorise files using them. Those who did use tags tended to only apply a single tag to each file or email. Bergman et al. concluded that the low use of tags, and the tendency to apply single tags when they were used, indicated that there is little enthusiasm for systems that support multiple classification. They provided potential reasons for this preference based on

participant comments, including that the extra effort to apply multiple tags was difficult, time consuming and redundant, that single classification was sufficient in most cases, and that multiple categorisation makes it less efficient to exhaustively explore the file system.

3.5.4 Organisation and Maintenance

Organisation and maintenance of file collections are important tasks that facilitate easier retrieval in the future. Naturally, people dedicate different amounts of effort into organising their hierarchies. This section begins by describing classifications of organisational strategies. It then discusses the difficulties of archiving information, attempts at automatic filing, difficulties in deciding when to delete information, and the role of annotations to supplement hierarchy classification.

Filers and Pilers

Malone [127] described two types of people based on their document management strategies for paper documents, later referred to as *filers* and *pilers* [176]. Filers are more organised, quickly classifying new documents and placing them in an appropriate location. Pilers spend less effort organising their documents, and their collections may appear to be less orderly. This reduced level of organisation means that it can be harder to remember document locations. Boardman and Sasse [34] created a similar classification, with *total filers*, who file most documents on creation, *extensive filers*, who file most items on creation but also manage a large set of unfiled items (typically classified as working or ephemeral information), and *occasional filers*, who leave most items unfiled, have few folders, and do not consider filing to be a priority. Most of Boardman and Sasse's participants were classified as total filers, with few occasional filers. Henderson [94] described another similar categorisation of document management strategies: *piling*, where the file hierarchy is broad and shallow, folders are used primarily for dumping large groups of old documents, and files accumulate in certain locations such as on the desktop; *filing*, where the hierarchy has medium width and depth and folders are created as needed; and *structuring*, where the hierarchy is narrow and deep

and folders are often created before they are needed. Similar classifications to these have also been noted for emails [185] and bookmarks [1].

Researchers have identified differences between pilers and filers in almost every aspect of file management, including acquisition of documents, organisation, maintenance and retrieval. Whittaker and Hirschberg [183] found that filers amassed more information than pilers. Filers were also more reluctant to discard information, due to the large amount of effort invested in organising it. Despite these larger document collections, filers accessed information less frequently than pilers. Teevan et al. [176] found that filers reported higher use of search than pilers, although they speculate that pilers likely underreported their search usage due to differences in definitions stemming from how the two groups normally use navigation. Henderson [94] found that those with structuring strategies (an extreme form of filing) found the lack of hierarchical context in search interfaces irritating, suggesting that search is less useful to filers. The different approaches of those with different organisational strategies suggests that a range of retrieval methods is necessary to satisfy everyone's preferences.

Archiving and Classification

File hierarchies develop over time through the process of classifying and archiving documents. While this is an important process to users [155], it has large overheads [183] and requires significant effort [155]. The most significant problem is the difficulty in determining the most appropriate location to store an item. Although people tend to organise in a way that facilitates easy retrieval [19], Whittaker and Hirschberg [183] found that people often forgot the categories they had already created, leading to duplicate categories that meant files were often overlooked when attempting to retrieve all the information on a topic. Errors made when classifying documents can result in items being far from their 'correct' locations [162]. Classification accuracy is made increasingly difficulty by changing information requirements that lead to ongoing changes in organisational structures [162, 155]. Unsurprisingly, users often feel that they have failed to effectively manage their hierarchies, impacting their self-image [35]. Barreau and Nardi [19] found that users of-

ten give up on elaborate filing systems because they do not yield sufficient value, with hierarchy structure more often determined in an ad hoc manner based on current tasks.

Ravasio et al. [155] also studied the process of classification. They found that classification was an ongoing process, with no structure considered permanent, but that maintenance efforts were most common following milestones such as at the end of a project. Subfolders were often created when 3-7 documents were related to the same topic, while the age of archived files typically ranged from six months to eight years. While differences in organisational strategy affect archiving behaviour [34], there are also differences based on the source of the information: self-created documents are normally archived immediately, whereas documents from an external source are more likely to be initially stored in temporary locations such as the desktop [155].

Automatic and Manual Filing

The previous section described the challenges users face in filing information. As a result of these difficulties, Malone [127] has suggested that automatic classification could ease this burden. Ravasio et al. [155] also found that users would like files to be filed automatically, albeit with the ability to adjust the result when they disagreed with it.

Sinha and Basu [167] describe Gardener, a system that suggests locations to save a file based on analysis of the filename entered by the user. However, this approach assumes that a filename would contain all the necessary information to enable this classification, when it is common for identifying information to instead be stored in the names of ancestor folders [108]. Barreau and Nardi [19] found that filenames were generally named for the purpose of reminding when scanning files at a particular location, rather than for use in search, suggesting that users are unlikely to specify all the information in a filename that would be required to fully classify the file.

Bao et al. [17] propose an alternate system that aims to minimise the number of navigation steps required when opening or saving a file by predicting the eventual folder. It relies on users first setting a current task, with task-specific folder predictions based on frequency and recency of past folder

use. In addition to using these predictions to set the default location for an open or save dialog, the system also provides several other folder predictions in case its top prediction is incorrect. A preliminary evaluation found that the system reduced navigation 'clicks' in open and save dialogs by about 50%.

Despite the potential benefits offered by these systems, automatic filing aids are not necessarily beneficial. Lansdale [121] noted that users have much stronger memory of the organisation that they have performed manually, commenting that users either have to file manually, creating a disincentive to do so and thus causing retrieval problems, or have filing automated, in which case users remember less and will also not be able to retrieve as well. To resolve this dilemma, automatic filing tools would need to find a mechanism in which they can facilitate retention of item locations in memory.

Keeping and Deleting

A difficult decision for users is often which information to keep, and which to delete [103]. This problem arises from the difficulty in evaluating the value of information and envisaging future needs [112]. Keeping information can be beneficial if it might be useful in the future, but deleting it reduces the overall size of the file hierarchy, potentially making it easier to retrieve other files. As the value of information decreases over time [183], information often needs to be re-evaluated.

Barreau [18] identified several common reasons for people to delete files. These include other versions of the document existing in another format, the user being finished with the document, the document being old, a lack of available disk space, new information becoming available, or having unknown content.

Bergman et al. [28] observed that files were often kept because users were hesitant about deleting them, even when they were unlikely to be useful in the future. To address this, they developed a system called GrayArea as a way of demoting unimportant files. Demoted items are displayed in a separate area at the bottom of the folder.

Annotations and Labels

Annotations are used in a variety of organisational systems. They can be used to aid search tools by providing additional properties to filter search results, or as visualisation aids to help find items within a folder.

Most modern operating systems, such as Windows and OS X, support comments and other manual annotations on files. OS X supports labels (Figure 3.1). These labels apply a background colour to files and folders when viewing them in the Finder, and allow for another dimension of organisation beyond hierarchies. As an example, a set of papers could be organised in a hierarchy based on topic, with labels used to indicate which have been read. Labels can be given custom names to facilitate this. OS X 10.9 adds the ability to apply multiple colour labels (renamed as 'tags') to items, allowing for further dimensions. Kaptelinin [112] found that this labelling feature was used to denote importance or project status, however it was not used extensively in practice.

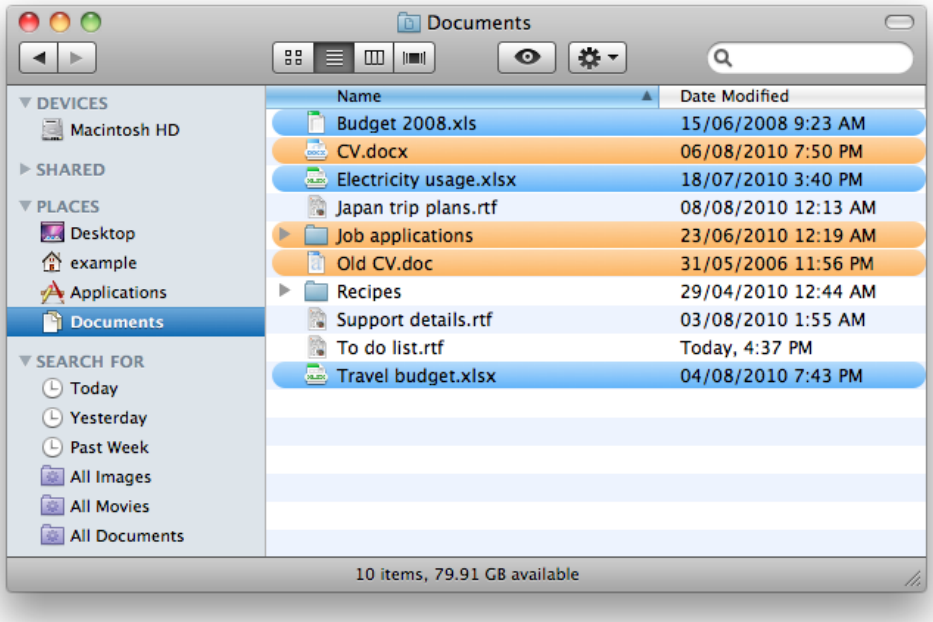


Figure 3.1: Labels in Mac OS X 10.6

Taking the definition of annotations to an extreme, they could include filenames and even locations themselves, as they are information manually added by users that are supplementary to the file content. Freeman and Gelernter [72] view the requirement of creating this data as unneeded overhead when creating files, noting that "when you grab a piece of paper and start writing, no-one demands that you bestow a name on the sheet or find it a storage location". Indeed, if annotations are used primarily to aid in finding files at a later time, filenames serve as a compulsory form of this.

The use of tags allow users to better express their own internal conceptualisation of file organisation, providing greater control and flexibility [44, 155]. However, while these annotations can be helpful for both organisation and retrieval, people seldom take the time to create or maintain them [133]. Bergman et al. [26] found that users considered tagging to be time consuming, and that it was slower to retrieve files using tags than navigating normally. Cruz and Xiao [52] found that users preferred computers to classify their information automatically rather than add manual annotations, even if the classifications were less accurate.

3.5.5 The Desktop and Spatial Locations

The desktop – as a metaphor for a physical desktop – serves an important role, providing permanent access to a customisable set of items, often at predictable spatial locations. This convenient access, the fact that users organise to facilitate easy retrieval [19], and the reminding function served by placing items so visibly [155], make the desktop a common location to store ephemeral and working information [112, 155, 19].

While system support for persistent spatial locations of icons within folders is mixed (see Table 2.2 in Chapter 2), the desktop supports user-customisable locations – either all the time or as an option – on all major systems. Users commonly make use of this feature to group related items together [155]. Ravasio et al. [155] found that users ordered and retrieved items spatially when there was support for it, while Dumais and Jones [62] found that names provided greater benefit to file retrieval, but that location information did provide limited utility. Moon and Fu [142] showed that users

compensate for a lack of spatial stability by using more of their memory on filenames, suggesting that interfaces that support consistent spatial locations – such as the desktop – can ease cognitive load.

The desktop does have some disadvantages [112]. As it exists below all other windows, its content is often obscured, and it cannot be brought to the front as easily as traditional windows (although some systems offer features to temporarily hide all windows or otherwise expose the desktop). Additionally, its permanent presence means that users are unable to close it to protect privacy, and this may impact which items they place on it.

3.5.6 Memory of File Attributes

Users have mixed memory of their files and those files' attributes. For example 10-17% of file locations are not remembered correctly [25, 80], and 8% of files are not recognised when prompted with their filenames [40]. Nardi et al. [145] found that users often remember where a file is but not its name. Memory of files is worse than that of folders, potentially because users pay more attention to naming folders than files [80].

Blanc-Brude and Scapin [32] ran a study to find which document attributes are most often recalled and how precisely they are recalled. They found that file type and visual elements within the documents could usually be recalled accurately, while size and date of last access were often recalled inaccurately. Results are summarised below, along with additional analysis on the relative usefulness of each attribute for file retrieval tools. These remarks make reference to precision and recall, common metrics in information retrieval [163]. Precision is a measure of the proportion of the result set that is relevant, while recall is a measure of how much of the relevant data in the search space is included in the result set.

Location – Locations were at least partially recalled 96% of the time, although only completely correctly 36% of the time. For the partially correct cases, the earlier portions of the paths were more likely to be remembered than the later portions. These findings suggest tools which have some leniency towards imperfect memory of file locations are likely to be more effective than those that require perfect recall.

Type or format – File types were correctly recalled 93% of the time, making them the most reliably recalled attribute. Participants often recalled high level types such as "a presentation" rather than low level types such as ".ppt". However, file types are not sufficient to uniquely identify files; Blanc-Brude and Scapin themselves stated that "[the] relative capacity of attributes to discriminate documents from each other and to express queries that will return small sets of results is also an important dimension to address in order to design efficient search tools". File types are therefore more likely to be useful as a supplementary filter to narrow down an initial set of results.

Filename – Filenames were at least partially recalled 92% of the time, but only completely correctly 25% of the time. An overwhelming majority of partial recalls consisted of a portion of the name, rather than erroneous text. This indicates that tools which can facilitate retrieval based on partial filenames are likely to be effective.

Title – Document titles were at least partially recalled 80% of the time, however partial recalls often contained erroneous portions of text (43%). Titles are therefore likely to help in locating files in many cases, but have low precision and recall relative to some other attributes.

Size – Document sizes (such as number of pages) were falsely recalled over 50% of the time, making size an unreliable attribute to use in file retrieval tools, and one which should have its use discouraged to prevent low recall.

Time – Like document size, the date that an item was last accessed was falsely recalled about half the time, suggesting its use as a filter should be deprioritised by file retrieval tools in order to encourage use of attributes with higher recall.

Keywords – Keywords, defined as meaningful words within a document, were recalled 32% of the time and partially the remaining 68% of the time. All partial recalls were the result of at least one recalled keyword

that was not contained in the document. This suggests that tools which use a naive 'AND' operator of search keywords are likely to have low accuracy. Additionally, examining the entire contents of each document results in a much larger search space for file retrieval tools, whereas filenames (for example) contain only a short amount of text. As a result, content searches are likely to have considerably more irrelevant results, and thus lower precision.

Visual elements – Visual elements of documents were the most reliably recalled attributes of documents after file types, with correct recalls 76% of the time and partial recalls the remaining 24% of the time. Unfortunately, due to the non-textual nature the the attribute, there are numerous problems with designing retrieval tools which use visual elements as the primary retrieval attribute. The result does suggest, however, that features such as thumbnails in retrieval results may facilitate selection of the correct file in the final stages of the retrieval process.

3.5.7 Retrieval Behaviour

The unique combination of properties of personal file collections, described in Section 3.1, mean that a range of retrieval methods can be used. Exhaustive and familiar hierarchies mean that navigation is a feasible method, while specialised methods such as 'Open Recent' menus are effective in many cases due to the influence of recency in file management [19]. Search is still important when hierarchy locations are unknown or cannot be correctly recalled.

Jones [104] described a four step process for finding items: (1) remembering to look; (2) recalling information about the item as input to a retrieval method; (3) recognising the desired item; and (4) repeating as needed for the set of items required. The second and third were introduced by Lansdale [121], however Jones notes that users often forget to perform tasks or reuse information (step 1), and that multiple items may be required for a single task (step 4).

Nardi et al. [145] noted user preferences towards navigation, and described the predominant retrieval pattern as looking in a location, looking

in a different location, then resorting to search tools. This section explores this preference towards navigation in the first instance, while also separately detailing findings related to the use of navigation and search.

Navigation Versus Search

Navigation- and search-based methods are the predominant retrieval methods that can be used to retrieve an arbitrary file. Substantial research has compared relative use of the two techniques, as well as the underlying cognitive reasons for user preferences. There is a general consensus that most users prefer navigation to search, with search being used only as a method of last resort [19, 32, 34, 145, 25, 155, 26, 104, 176]. However, search still offers important benefits in certain situations.

Early studies on search interfaces were based on primitive systems that were slow to return results, produced inaccurate results, or could only match results based on limited metadata, such as filenames. These technological obstacles were often cited as the cause of low usage rates, with speculation that improvements might improve adoption (e.g., [145, 155]). Bergman et al. [25], however, showed that improvements to search systems had little effect on perceptions or usage rates. In a survey of Windows and Mac users, they found a strong preference towards navigation over search regardless of the sophistication of the underlying search tool. They concluded that search was used mainly as a last resort, and was rarely used when the location of a file was known.

Bergman et al.'s findings are explained by the relative cognitive requirements of the two techniques. Users prefer orienteering (that is, taking small steps towards a target using partial information and contextual knowledge) to teleporting (that is, jumping directly to the target) [155, 176]. Navigation uses an orienteering approach, with users able to use recognition at each step of a retrieval to identify the next folder [25]. Orienteering offers several advantages over keyword search, including decreased cognitive load, a sense of location, and a better understanding of the result [176]. Bergman et al. [25] also note that, with navigation, "users can continue to think of the project they are working on at the time", even if search might be faster.

They conclude by stating "perhaps it is time to explore alternative [Personal Information Management] design directions which focus on navigation and the improvement of human computer interface based on users' perceptions and preferences".

Search interfaces, on the other hand, typically use a teleporting approach that shows an immediate list of results with little or no context [176]. Search also relies on users recalling attributes of a target file in order to devise a search query [19, 25], which is more cognitively demanding than recognition [179]. Furthermore, search offers no reminding feature. This means that users are unlikely to encounter an item through search if they have forgotten they have it or how it is described in the file system, resulting in a lower sense of control [19]. A final potential limitation of search-based file access is that it provides minimal support for learning and rehearsing the location-based retrieval mechanics that users are likely to use for future accesses. Search queries either succeed (and yield the desired target in a small set of candidates) or they fail. When successful, the user is likely to proceed working with the file rather than study and memorise the location, and consequently when the file is next required the user's location knowledge will not have improved.

These differences between search and navigation have been confirmed by other studies. Bergman et al. [27] showed that users were better at performing a secondary task while navigating than when using search, demonstrating that navigation is less cognitively demanding. They also found that search was slower than navigation and more vulnerable to failure. Ravasio et al. [155] note that users are more likely to perform manual searches by exhaustively exploring their hierarchy using navigation. Their participants would first attempt a *Direct Access Strategy*, followed by exploring nearby folders in a logical order, then exhaustively searching all possible locations. While these participants acknowledged that this was often slow, it was generally successful. Search interfaces were only used when participants were close to considering information lost or had no indication of a file's location, but these interfaces were both cognitively and mechanically demanding.

Navigation

Several studies have examined the use of navigation. In a large-scale study with 289 participants, Bergman et al. [30] examined the effect of operating system, view and hierarchy depth on navigation retrievals. Their method involved statically recording the state of participants' recent documents list, then asking them to navigate to each of those files using a file browser, while video captured their actions. By analysing the video they found that Mac and Windows users structured their files in different ways, with Windows users containing more files in each folder than Mac users, but fewer subfolders. As a result, retrieved files were deeper in the file hierarchy on Windows (2.9 levels deep, compared to 2.4 levels on Mac OS X), and file retrieval times were slower (17.3 seconds on Windows, 12.6 seconds on Mac OS X). They also found that on both Windows and Mac OS X, icon view had the lowest step durations (the time spent at each level of the hierarchy) of all the available view types. Finally, they found that folder depth did not affect step duration, which contrasts Laundauer and Nachbar's findings [120] of hierarchical menu traversal, which showed lower step times when selecting a hierarchy leaf.

In an earlier analysis of the same data, Bergman et al. analysed the effect of folder structure on navigation [29]. They found that participants could successfully locate 94% of their recent documents, though 21% of these retrievals included a navigational error en route to the file. Folders at higher levels of the hierarchy contained both more files and subfolders; overall, folders contained a mean of 10.6 subfolders and 11.8 files. 12% of all folders contained only files, and 20% contained only folders. They also found that errors were more likely in larger folders, as well as for files deeper in the file hierarchy, and developed a predictive model for retrieval time based on the file depth and average folder size.

Others have also examined the use of particular file browser views. In a 1995 Mac study, Nardi et al. [145] found that icon view was common at the top level of the disk when users could set spatial locations. At deeper levels, it was less common, but was used when icons or thumbnails were important. Alphabetic list views were used in folders with many similar files. On Windows, Golemati et al. [80] found that a tree view, featuring expand-

able folders but omitting files, was not used by most users, and users disliked having to switch attention between it and a separate view that showed the content of the selected folder.

Other findings on navigation include that files stored deeper in the hierarchy require more cognitive effort to retrieve [27] and that the use of unique icons for each file and folder decreases retrieval times even when the icons have no relation to the items they represent [123].

Search

Despite preferences towards navigation, search remains an important tool for file retrieval, particularly when item locations are unknown [25]. Search has many attractive features for file retrieval: any file attribute can be searched, rather than requiring memory of the item's location [121]; it does not depend on a hierarchy, relieving users of the need to recall semantic groupings; and it enables retrieval in a single step [25], potentially allowing for faster retrievals in some cases.

Many of the limitations of search arise from the lack of context provided by most search interfaces [176]. Soules and Ganger [169] attempted to partly alleviate this by combining both content and context analysis to produce file search results. Context analysis was performed using a graph of temporal file access relationships. This combined approach triggered improvements to both recall and accuracy of search results. Hearst [91] also advocated the inclusion of context in search results by organising results into meaningful groups. Käki [111] found that categorising web search results proved beneficial, making it easier to find poorly ranked items and allowing for simpler queries.

3.6 Conclusion

This chapter provided a review of existing organisation and retrieval literature. While the focus was on file management, it began with a comparison of the properties of several domains in which items are regularly retrieved, providing insights into how these property differences affect the ways in which people interact with their information. This was followed by brief summaries

of the management of web bookmarks, emails and paper documents, providing context for file management.

The remainder of the chapter detailed file management literature. To summarise, files are stored in a hierarchy structure where files have a single unique location. While there are many conceptual advantages to files existing in multiple locations (e.g., [57]), there is little enthusiasm for such systems [26]. People interact with ephemeral, working and archived information [145], organising and retrieving them in different ways. There are a range of document management strategies, with people often being classified as either filers or pilers based on their tendency to organise information [176]. People have good, but not perfect memory of file locations and names [32], and prefer to navigate to files rather than search for them, due to the reminding features offered by navigation interfaces [19].

This review, combined with the overview of file retrieval methods in Chapter 2, provides an important initial review of file retrieval in the larger context of Personal Information Management. The following chapters expand this understanding to further characterise the ways in which people retrieve their files.

Part II

Characterising File Retrieval

Chapter IV

FileMonitor: A Tool To Understand File Retrieval Behaviour

A thorough understanding of how people use existing file retrieval tools is important to designing improved tools. Such information tells us what different types of retrieval behaviour users exhibit, what features and methods are most often used, and what inefficiencies exist that can be remedied with better interface design.

The domain of file retrieval exhibits several challenges in achieving external validity in studies aimed at gaining this understanding. File retrieval involves large, familiar file hierarchies that are difficult to simulate accurately in a lab study, and there are a range of retrieval techniques that have complex interactions with each other that influence their suitability for a particular task.

Numerous studies have previously attempted to gather data on file retrieval or organisation. Several studies have used snapshots of file hierarchy organisation [82, 95, 173], however such data cannot be used to describe file *retrieval*, or to provide any temporal understanding beyond that provided by file metadata. Others have involved questionnaires or interviews [25, 145], but such approaches are limited in the amount of data they can provide for the purposes of detailed analysis, and actual usage behaviour may not match user estimates. Bergman et al. [29] recorded video of participants retrieving recent files on their own computers, but recording and analysing sufficient footage to analyse long-term patterns of behaviour is not feasible, and participants may behave differently when they know they are being filmed.

The limitations of these approaches suggest that full external validity may be best achieved by observing user behaviour using a logging tool. While such

tools are challenging to implement, they can silently monitor user behaviour over an extended period without interfering with the way users use their computers.

Previous tools have been developed to log user behaviour in other contexts. Software from major vendors often includes features, such as Microsoft's *Customer Experience Improvement Program*, that automatically report usage information to the vendor. These tools have the advantage of integration within the main codebase, allowing for simpler implementation and increased stability, but results gathered from these tools are rarely disclosed publicly. Researchers have also developed tools that run independently of the software they are observing. For example, AppMonitor [6] observes application usage of Microsoft Word and Adobe Reader, and PyLogger [172, p. 37] observes window switching behaviour on Microsoft Windows. The authors highlight the need for these tools to maintain existing system performance and stability; such properties are important both because slow or unstable software might affect how participants use their computers, and because they allow participants to forget their are being observed, further minimising any changes in user behaviour.

This chapter describes *FileMonitor*, a tool that monitors file retrieval behaviour on OS X. Like AppMonitor and PyLogger, it observes user behaviour without requiring users to change the software that they use, and logs behaviour without noticeably affecting system performance or stability. The description of FileMonitor is provided in order to assist other researchers in developing similar tools in the future – a task that is made difficult by the lack of standard APIs or methods to monitor other applications.

The chapter first provides an overview of FileMonitor and the mechanisms it uses to observe file retrieval behaviour. Next, it discusses the implementation of FileMonitor, along with the associated challenges of developing it. Finally, it describes the logs that FileMonitor generates, including specific details of the possible types of log entries.

4.1 An Overview of FileMonitor

FileMonitor is a plugin for the Finder, the default file browsing application on OS X. It silently monitors and logs user behaviour, while remaining invisible to users.

FileMonitor records three types of behaviour:

1. **Finder usage**: file browsing behaviour within the Finder, such as file browser window management, opening folders, opening files and performing searches. This is achieved by intercepting method calls in the Finder runtime to trigger FileMonitor's logging code.

2. **Spotlight usage**: use of Spotlight, OS X's search technology. FileMonitor records search terms and which search results are selected in the system-wide Spotlight menu, using OS X's accessibility API.

3. **Other file open events**: any file opened using any method other than those above, such as open dialogs, "Open Recent" menus or third party tools. In such cases, it can only be determined *that* the files were opened, not *how* they were opened. To record this information, FileMonitor observes changes to the system's record of recent documents.

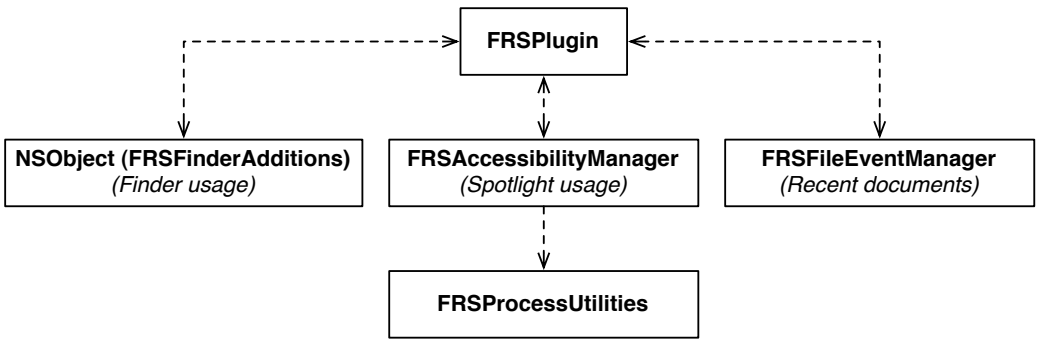


Figure 4.1: Simplified class diagram of the FileMonitor plugin

4.2 FileMonitor Implementation

FileMonitor was implemented as a SIMBL [168] plugin for the Finder, implemented in Objective-C. It is injected into the Finder on launch, and is compatible with Mac OS X 10.6 (Snow Leopard) and 10.7 (Lion).

A simplified class diagram of the plugin is shown in Figure 4.1. The *FRSPlugin* class sets up the various logging components, and also handles the logging itself. Details of the logging implementation are described in Section 4.2.1. *FRSFinderAdditions* is a category of *NSObject* (Cocoa's root object class) and monitors Finder usage, described in Section 4.2.2; note that an Objective-C category is an extension of an existing class that adds functionality to all objects of that class. *FRSAccessibilityManager* monitors Spotlight usage using the Accessibility API, described in Section 4.2.3, and uses *FRSProcessUtilities* to get information about the frontmost process. *FRSFileEventManager* monitors the system's recent items file, described in Section 4.2.4.

4.2.1 Logging

All logging is handled by the *FRSPlugin class*, with logging methods called by other components. Log files are created and stored locally on the user's computer. One file is created for each day that FileMonitor is running.

Log files are in plain text format encoded in UTF-8, and contain one line for each log entry. Each line is formatted as follows:

[date/time] [event name] Params: *[param1] [param2]*, Windows: *[window1] [window2]*, View: *[view]*

The two event parameters *param1* and *param2* are enclosed in quotes, and any quote or backslash characters within the parameter are escaped with a backslash. In some cases, the parameters are lists, in which case the list is formatted as a comma separated list enclosed in brackets, with each component enclosed in quotes as above. The window and view parameters are specific to Finder usage. When a parameter is not specified, it is logged as "0", or "Unknown" for the view parameter.

4.2.2 Finder Usage

The majority of logging in FileMonitor occurs as a result of actions performed in the Finder. This is achieved using a technique known as method swizzling [48], which uses the Objective-C runtime to substitute an existing method implementation for another – in FileMonitor, this is used to substitute existing method implementations in the Finder with custom implementations. Typically, FileMonitor's implementations simply log the event, then call the Finder's original implementation, so that the net effect is an injection of logging code.

Method Discovery

Swizzling Finder's code required some knowledge of the internal structure of Finder's architecture, as well as the specific methods in each of its classes. Several tools helped to gain this information.

- **class-dump** [171] generates header files for Mach-O files, such as the Finder executable file. The Finder header files provide information about the structure of the Finder, i.e. the classes it contains and the methods contained within them. Except for primitive types, it does not provide information about the types of method parameters or return values.

- The **instrumentObjcMessageSends()** function enables logging of every Objective-C message sent (Objective-C messages are similar to method calls in other languages). Enabling it before a method is called, then disabling it immediately after, provides some information about the internal behaviour of the Finder, especially when combined with swizzles of low level methods such as mouse down event handlers. These log files can be extremely long, however, and tend to only provide clues for further exploration. They are also limited to Objective-C messages and do not include C function calls.

- **F-Script** [152] can inject itself into applications and provides a hierarchical browser of the views within an application's interface. It can

inspect specific instantiations of these views to examine property values, providing further information as to what properties and methods are of relevance.

Even with these tools, large amounts of experimentation were required, especially when dealing with proprietary data types. In many cases, separate code had to be used on Mac OS X 10.6 and 10.7, due to internal changes in the Finder between releases. This is a common problem in the development of logging tools, and some pootential solutions are discussed in Section 4.4.

Finder Usage Log Content

Details of all the log events are included in Section 4.3. Each entry includes some common information:

- A unique identifier of the source window for the event, allowing window use to be tracked over time. Some events also log identifiers for new windows that are created in response to the event, such as opening a folder in a new window. For the purposes of logging, the desktop counts as a window, and log entries are created when the desktop window is first "opened" (i.e., on launch).

- The type of view in the source window, where applicable. Examples include icon view, list view, column view and the desktop.

4.2.3 Spotlight Usage

The Spotlight menu in OS X is controlled by the *SystemUIServer* process, which cannot have code injected into it in the same way as the Finder. Instead, the OS X Accessibility API [8] provides access to its interface elements. This functionality is enabled when the "Enable access to assistive devices" option is set in the *Universal Access* system preference pane.

Observation

To record Spotlight events, FileMonitor creates observers that fire notifications when certain events are triggered. For the purposes of event logging,

these events are search query changes and results selections.

Search query changes are observed with a *value changed* observer attached to the search field. However, this observer cannot be created when FileMonitor is launched, as the Spotlight menu needs to be active to retrieve a reference to the search field. Instead, FileMonitor creates a system-wide observer for key down and mouse up events which checks Spotlight menu visibility. Once it observes that the menu has been activated, it adds a *value changed* observer to the search field, which remains attached until the process is restarted.

To detect results selections, FileMonitor adds application-wide observers to *SystemUIServer* on launch. On Mac OS X 10.6, selecting a result (i.e., clicking it or using the keyboard to open an item) is detected by observing *menu item selected* events. However, the Spotlight menu does not trigger these events on Mac OS X 10.7. As a workaround, FileMonitor also registers observers for *menu selection change* and *menu close* events. When the former event is triggered (e.g., by hovering over a menu item so as to highlight it, but before clicking it), FileMonitor records the path of the file represented by the menu item, as well as the identifier of the frontmost application. When the menu is closed, after a small delay, FileMonitor compares the new frontmost application to the most recently recorded one. If they are different, it is assumed that the previously recorded path was selected, rather than the menu being closed without making a selection. If they are the same, it is assumed that no selection was made. While this would not be true in all cases (i.e., if a search result is opened in the application that is already active), it is likely to be generally true. This method is only used as a fallback when no *menu item selected* events are triggered.

Spotlight Log Content

When Spotlight search text changes, FileMonitor logs *SpotlightSearch* events, which record the new search text. When a result is selected, it logs *SpotlightSelection* events, which record the path of the selected item.

4.2.4 Recent Documents

Given the large number of methods that can be used to open files, many involving third party software, it was impractical to write specific logging code for each. However, OS X maintains a file that lists recently opened documents, applications and servers. FileMonitor can determine which files have been opened by monitoring changes to this file, stored at *~/Library/Preferences/com.apple.recentitems.plist*, subject to limitations described below.

The recent items document contains separate ordered lists of recent documents, applications and servers. Items that occur earlier in the lists have been accessed more recently. The total number of items in each list is capped at a level determined by settings in the *Appearance* (Mac OS X 10.6) or *General* (Mac OS X 10.7) preference panes, but defaults to 10 each. In order for FileMonitor to be able to determine recent items, these settings must not be set to 'None', however it is rare for users to select this option.

Files are only recorded in this file if they are opened in response to user events. For example, a settings files would not be recorded if it was opened in the background rather than in response to a direct user request, however it would if it were opened using an "Open" dialog. In some cases, it is impossible to distinguish between a file being opened, and the window of an already-open document being activated, which are also recorded in some applications. There are also some other variations between applications; for example, in the *Xcode* IDE, switching between source code files within a project would cause a file to be recorded on each switch. In the *Mail* application, however, opening emails (represented as individual files on disk) is not recorded. As no details apart from the file path are recorded in the recent items file, FileMonitor cannot detect this application-specific behaviour.

When a file is opened, it is not immediately recorded in the recent items file. Instead, there is a delay of up to a few minutes before it is updated. The recent items file is up-to-date when updated, so a single change in the file often corresponds to multiple file open events.

Observation

To monitor changes in the recent items file, FileMonitor sets up a file system event stream (FSEventStream) for the file, which notifies it of any changes to the file. As the file does not record times that the files were opened, FileMonitor records a copy of the recent items on its initial launch, and from then on compares the lists of recent documents and applications to the lists from the previous update to deduce which files have been opened. Recent documents are processed separately from recent applications.

Lists of documents and applications are stored in descending order of the time of last access. When comparing lists, items are considered new until reaching two consecutive items that appear in the same order in the previous list, with intermediary items allowed in the previous list only if they have already been observed in the new list. This guarantees no false positives, although false negatives are possible in rare cases. Roughly, the algorithm works as follows:

1. For each item I in the new list:
   (a) If I was in the previous list, there is a held item H, and I appears in the previous list immediately after H (excluding any intermediary items that have already been accepted), reject both and skip to step 2.
   (b) If there is a held item, accept it.
   (c) If I was not in the previous list, accept it and continue to the next item.
   (d) Otherwise (if I appears in a previous list), hold I.
2. Process accepted items in reverse order.

Table 4.1 illustrates an example, with comments for what action is performed for each item in the new list. In this case, the first three items are accepted, and processed in the order *Bag*, *Hand*, *Car*.

As the recent items file includes items opened through any method, it compares accepted items to those directly observed through Finder or Spotlight search. Any files that were part of directly observed retrievals since the

Previous list:	Apple, Bag, Car, Dog, Egg, Frog
New list:	Car, Hand, Bag, Apple, Dog, Egg

New Item	**Action**
Car	Held (appears in previous list), later accepted
Hand	Accepted (does not appear in previous list)
Bag	Held (appears in previous list), later accepted
Apple	Held (appears in previous list), later rejected
Dog	Rejected (appears after *Apple* in previous list, with only accepted items in between)
Egg	Not examined

Table 4.1: Example recent items list comparison

last change to the recent items file are ignored, since they have already been logged.

Recent Documents Log Content

Documents and applications from the recent items file are logged as *OpenFileExternal* events, which include the path of the file. The event time is when the file was processed, with may be delayed up to several minutes from when the file was opened.

4.3 FileMonitor Logs

FileMonitor produces a variety of log events. Table 4.2 lists the names of all log events, along with a description of each. Table 4.3 lists actions from a *user* point of view, with details of the log events that are triggered by these actions. Finally, Listing 4.1 provides an example excerpt from a FileMonitor log file.

Log Event	Description
	Opening items
OpenFile	A file has been opened in response to explicit user interaction within the Finder.
QuickLook	A file has been previewed in a popup window using OS X's "Quick Look" feature.
OpenFileExternal	A file has been opened in an application other than the Finder, as described in Section 4.2.4. These entries usually occur a minute or two after the file has been opened.
OpenFolder	The current path of a file browser window has changed. This can have a number of causes, such as opening a folder, opening an enclosing folder, selecting a folder from the sidebar or toolbar, or using back and forward buttons. The second parameter is used to distinguish between these (and other) cases.
OpenFolderNewWindow	A folder has been opened in a new window, rather than changing the location of an existing one. This has similar causes to *OpenFolder* and is sometimes done by holding down modifier keys when performing an action. For these events, identifiers for both the source window (where the event was triggered) and new window (created in response to the event) are provided.
CloseFile	A file has been closed. These entries can only be logged in Mac OS X 10.7, and only when the folder that contains the file is still open, so is of limited use.
	Window management
NewWindow	A new Finder window has been created, with or without a direct request for a specific path, but not due to a specific action in another window (in which case an *OpenFolderNewWindow* event will occur instead). Causes include the user using the "New Finder Window" command, a window being opened on launch, a search window opening, or a location being selected from the "Go" menu. The second parameter is used to distinguish between these cases.
CloseWindow	A Finder browser window has been closed.
	Other file tasks
RenameFile	A file has been renamed. Parameters include the old and new path. Rename events occur when renaming a file in a browser window or info/inspector window.
RenameFolder	Identical to *RenameFile*, except when applied to a folder.
NewFolder	A folder has been created as a result of direct user interaction. The parameter is the initial path of the folder, before renaming, such as "untitled folder". As such, this event is often coupled with a *RenameFolder* event.
Move	Files or folders have been moved as a result of direct user interaction, such as drag and drop. Parameters provide both the old and new path or paths.
Delete	Files or folders have been moved to the trash as a result of direct user interaction, such as drag and drop or with the "Move to Trash" command. No events are logged if items are later moved out of the trash, or when the trash is emptied.

Continued on next page

Log Event	Description
SetLabel	A colour label has been changed for an item, including the removal of a label. Parameters provide the path of the item and an identifier for the new label.
	Search
SearchScope	The scope of a search has changed, or an initial scope is determined. This event occurs when a new search is created or when the users changes whether the search is to take place across the whole computer of just within the current location.
Search	A search parameter has changed, including the primary query or any other search criteria. The text query is included with this event.
SearchQuery	A search query has changed. Similar to *Search*, but provides the raw Spotlight query provided to the Spotlight engine. This query includes all search criteria, beyond just the primary query. *SearchQuery* events will always immediately follow *Search* events and correspond to the same query.
SearchCancelled	A search has been explicitly cancelled, for example by clicking the search field's cancel button. Navigating away from the search, for example by clicking on a folder in the sidebar or using the "Back" button, will not trigger a *SearchCancelled* event. While cancelling a search often returns the window to its previous target, in some cases it just results in an empty search (such as when the window was created as a search window). In these cases, a new search may not be preceded by an initial *SearchScope* event as is typical of most searches.
SearchSaved	A search has been saved as a *saved search*. Parameters specify the search name and whether it was added to the sidebar.
OpenSavedSearch	A saved search has been opened in a browser window. This will generally be followed by a *SearchScope* event after the search is loaded.
SpotlightShowAll	The "Show All in Finder" command has been selected in the Spotlight menu after performing a search. This is generally preceded by a *NewWindow* event corresponding to the created results window in Finder.
SpotlightShortcut	A new search window has been created in response to the system-wide Spotlight window keyboard shortcut. As with *SpotlightShowAll*, this is generally preceded by a *NewWindow* event.
SpotlightSearch	The text in the Spotlight menu's search field has been changed. Usually occurs multiple times in succession, as it produces a log entry for each key press.
SpotlightSelection	A search result has been opened from the Spotlight menu. This will typically be immediately preceded by a SpotlightSearch entry, which provides the search query.

Table 4.2: FileMonitor log event descriptions

User Action	Log Event	Parameters	Windows
Opening files			
Open single file	OpenFile	1: *File path*	1: Source
Open multiple files	OpenFile	1: *File path array*	1: Source
"Open With"	OpenFile	1: *File path/array* 2: OpenWith	1: Source
"Open With Other"	OpenFile	1: *File path/array* 2: OpenWithOther	1: Source
Open file from toolbar	OpenFile	1: *File path* 2: Toolbar	1: Source
Open file from search (Open with/Open with other)	OpenFile	1: *File path* 2: Search (Search-OpenWith /Search-OpenWithOther)	1: Source
Quick Look file	QuickLook	1: *File path*	1: Source
Opening file in other application	OpenFileExternal	1: *File path*	N/A
Close file (10.7 only)	CloseFile	1: *File path*	N/A
Opening folders			
Opening in same window	OpenFolder	1: *Folder path*	1: Source
Opening in new window	OpenFolderNewWindow	1: *Folder path*	1: Source 2: New
Open enclosing (same window)	OpenFolder	1: *Folder path* 2: Enclosing	1: Source
Open enclosing (new window)	OpenFolderNewWindow	1: *Folder path* 2: Enclosing	1: Source 2: New
Opening from sidebar (same window)	OpenFolder	1: *Folder path* 2: Sidebar	1: Source
Opening from sidebar (new window)	OpenFolderNewWindow	1: *Folder path* 2: Sidebar	1: Source 2: New
Opening from toolbar	OpenFolder	1: *Folder path* 2: Toolbar	1: Source
Show package contents	OpenFolderNewWindow	1: *Package path* 2: PackageContents	1: Source 2: New
Open recent folder (in new window)	OpenFolder (NewWindow)	1: *Folder path* 2: Recent	1: Source
Back	OpenFolder	1: *Folder path* 2: Back	1: Source
Forward	OpenFolder	1: *Folder path* 2: Forward	1: Source
Show original	OpenFolder	1: *Folder path* 2: ShowOriginal	1: Source
Go to folder (in new window)	OpenFolder (NewWindow)	1: *Folder path* 2: Goto	1: Target
Go to specific folder *X* (in new window)	OpenFolder (NewWindow)	1: *Folder path* 2: GoTo*X*	1: Target
Other file tasks			
Rename file	RenameFile	1: *Initial path* 2: *New path*	1: Source
Rename folder	RenameFolder	1: *Initial path* 2: *New path*	1: Source

Continued on next page

User Action	Log Event	Parameters	Windows
New folder	NewFolder	1: *New path*	1: Source
Move item(s)	Move	1: *Old file path/array* 2: *New file path/array*	1: Source
Delete item(s)	Delete	1: *File path/array*	1: Source
Set label	SetLabel	1: *Label identifier*	1: Source
Window management			
New window	NewWindow	1: *Initial path*	2: New
Initial window on launch	NewWindow	1: *Initial path* 2: Initial	2: New
Close window	CloseWindow	*None*	1: Source
Search			
Search scope change	SearchScope	1: *Scope*	1: Source
Search changed	Search	1: *Search text* 2: Contents/Filenames	1: Source
	SearchQuery	1: *Spotlight query*	1: Source
Search cancelled	SearchCancelled	1: *New path*	1: Source
Save search	SearchSaved	1: *Saved search path* 2: *Added to sidebar?*	1: Source
Use saved search	OpenSavedSearch	1: *Saved search path*	1: Source
Show all (Spotlight)	NewWindow	1: N/A 2: Spotlight	1: New
	SpotlightShowAll	1: *Search query* 2: New/Existing	1: New
New Spotlight search (in Finder)	NewWindow	1: N/A 2: Spotlight	1: New
	SpotlightShortcut	1: N/A 2: New/Existing	1: New
Search text changed (Spotlight menu)	SpotlightSearch	1: *Search text*	N/A
Result selected (Spotlight menu)	SpotlightSelection	1: *File path*	N/A

Table 4.3: FileMonitor log event details

4.4 Discussion

This chapter described FileMonitor, a tool that monitors file retrieval behaviour on OS X. Logging tools such as FileMonitor help to provide valuable information about usage behaviours, and the description of FileMonitor included details of three different data-collection techniques which may be of use to researchers designing similar tools.

Unfortunately, logging tools typically require significant engineering resources due to the difficulties involved in observing existing software. Including support for multiple versions of the observed software can be particularly challenging, as logging tools often have to rely on the internal architecture

```
2012-06-03 10:28:28.902 CloseWindow, Params: 0 0, Windows: 124064 0, View: Column
2012-06-03 10:28:46.187 NewWindow, Params: "/" 0, Windows: 0 127255, View: Column
2012-06-03 10:28:47.260 OpenFolder, Params: "/Videos" 0, Windows: 127255 0, View: Column
2012-06-03 10:28:49.210 OpenFolder, Params: "/Videos/Unwatched" 0, Windows: 127255 0, View:
     Column
2012-06-03 10:29:05.899 Delete, Params: "/Videos/Unwatched/Keynote.mov" 0, Windows: 127255
    0, View: Column
2012-06-03 10:38:07.913 CloseWindow, Params: 0 0, Windows: 127255 0, View: Column
2012-06-03 10:38:17.505 Move, Params: "/Users/home/Downloads/handbook13-optimized.pdf" "/
    Users/home/Downloads/Unsorted/handbook13-optimized.pdf", Windows: 124038 0, View:
    Column
2012-06-03 10:38:19.764 OpenFile, Params: "/Users/home/Downloads/googlechrome.dmg" 0,
    Windows: 124038 0, View: Column
2012-06-03 10:38:21.752 NewWindow, Params: "/Volumes/Google Chrome" 0, Windows: 0 127426,
    View: Icon
2012-06-03 10:38:26.050 OpenFile, Params: "/Volumes/Google Chrome/Google Chrome.app" 0,
    Windows: 127426 0, View: Icon
2012-06-03 10:38:50.601 CloseWindow, Params: 0 0, Windows: 127426 0, View: Icon
2012-06-03 10:39:00.494 Delete, Params: "/Users/home/Downloads/F-Script.ML.universal.zip"
    0, Windows: 124038 0, View: Column
2012-06-03 10:39:12.182 Delete, Params: ("/Users/home/Downloads/CHI-paper-format-LaTeX", "/
    Users/home/Downloads/CHI-paper-format-LaTeX.zip") 0, Windows: 124038 0, View: Column
2012-06-03 11:21:37.856 NewWindow, Params: "/" 0, Windows: 0 127767, View: Column
2012-06-03 11:21:39.198 OpenFolder, Params: "/Users/home/Documents/Programming/Projects/
    SquiggleDemo" "Sidebar", Windows: 127767 0, View: Column
2012-06-03 11:21:41.458 OpenFolder, Params: "/Users/home/Documents/Programming/Projects/
    SquiggleDemo/Files" 0, Windows: 127767 0, View: Column
2012-06-03 11:21:44.662 OpenFolder, Params: "/Users/home/Documents/Programming/Projects/
    SquiggleDemo/Files/Notes" 0, Windows: 127767 0, View: Column
2012-06-03 11:21:52.284 OpenFolder, Params: "/Users/home/Documents/Programming/Projects/
    SquiggleDemo/Files/Miscellaneous" 0, Windows: 127767 0, View: Column
2012-06-03 11:21:55.760 Move, Params: "/Users/home/Downloads/linedraw.m" "/Users/home/
    Documents/Programming/Projects/SquiggleDemo/Files/Miscellaneous/linedraw.m", Windows:
    124038 0, View: Column
2012-06-03 11:21:59.943 OpenFile, Params: "/Users/home/Documents/Programming/Projects/
    SquiggleDemo/Files/Miscellaneous/linedraw.m" 0, Windows: 127767 0, View: Column
2012-06-03 11:30:52.268 RenameFile, Params: "/Users/home/Documents/Programming/Projects/
    SquiggleDemo/Files/Miscellaneous/linedraw.m" "/Users/home/Documents/Programming/
    Projects/SquiggleDemo/Files/Miscellaneous/SDLine.m", Windows: 127767 0, View: Column
2012-06-03 11:30:56.319 QuickLook, Params: "/Users/home/Documents/Programming/Projects/
    SquiggleDemo/Files/Miscellaneous/fractal.m" 0, Windows: 127767 0, View: Column
```

Listing 4.1: Sample excerpt from a FileMonitor log file

or behaviour of the software they are observing, and this will often change between releases.

Major software vendors could better support the research community in two ways. First, they could provide information gathered from their own reporting tools to researchers. Although this data is generally anonymised, they could further protect privacy by providing an API through which analysis can be performed at an aggregate level, without exposing data from any particular individual. Second, operating system vendors could include standard

APIs that provide extensive support for third party developers to monitor user interaction performed in other applications. While limited APIs are already available (for example, the Accessibility API on OS X), these could be expanded significantly. To alleviate security concerns, the system could ask for user permission to allow an application to make use of this API. These changes would be beneficial both for the research community, as well as for the vendors themselves; for example, it would allow vendors to utilise the findings of external researchers in order to improve their products, reducing research and development costs.

4.5 Conclusion

This chapter described the design and implementation of *FileMonitor*, software that silently monitors and logs file retrieval behaviour. By running as a plugin to OS X's file manager, the Finder, FileMonitor is always running, but is invisible to the user. This enables it to monitor retrieval behaviour without affecting it, resulting in a high degree of external validity.

FileMonitor uses three mechanisms to observe user behaviour: injecting code into the Finder to observe specific user actions, such as navigating through the file hierarchy and opening files; using OS X's accessibility API to observe searches within the system-wide Spotlight menu; and monitoring changes to the system's recent items file to deduce files that have been opened with other methods. This range of techniques provides both a broad overview of all file retrieval activity across the system, as well as detailed activity of files retrieved with specific methods (i.e., navigation and search).

While FileMonitor provides comprehensive logs of file retrieval behaviour, it is unable to provide context around them, such as *why* users choose to retrieve files in a particular way. Combining FileMonitor with other techniques, such as user interviews, can fill this gap and provide a detailed understanding of file retrieval behaviours and motivations. Chapter 5 reports the results of a study that uses both of these techniques, with FileMonitor installed on participants' computers for a month, and follow-up interviews that provide context for the content of the logs. Such characterisation is an important aid to the informed design of next-generation file retrieval interfaces.

Chapter V

How Do Users Retrieve Files? An Empirical Characterisation of File Retrieval

Conducting work on computer systems is normally preceded by an explicit user action to open or retrieve a file, such as a word processing document, a spreadsheet, or PDF document. Unsurprisingly, for an action that is a critical prerequisite to accomplishing tasks, many alternative tools are available to assist users with file retrieval. These tools often form an elemental part of the operating system, including hierarchical file browsers that support user controlled traversal of the file system (such as Windows' File Explorer and the OS X Finder), tools to view recently accessed files, and search utilities such as OS X's Spotlight. Individual software applications also provide tools for accessing files, normally through an "Open" command or using application-dependent recent file lists. Finally, third party vendors also offer tools for assisting file retrieval, such as specialised application launchers.

As expected for such an important user activity, there has been extensive research into how people manage their file systems and retrieve their files (reviewed in Chapter 3). The primary purpose and utility of this previous research is that it provides an understanding of current activities and patterns of behaviour that reveal opportunities for improving file retrieval performance in next generation interfaces – for example, they show bottlenecks in performance and can highlight frequently repeated activities that could benefit from new tools.

While the results of studies to date provide a good introduction to file retrieval activities, their findings are limited due to the necessary constraints imposed by their experimental methodologies. Specifically, most prior studies have one or more of the following limitations: (1) *temporal snapshot* –

they involve a very limited time period of analysis, often a single experimental session; (2) *out of context* – the file retrieval activities are analysed in response to specific requests from the experimenter, rather than as they arise in response to work requirements; and (3) *small scope* – direct observation and controlled experiments are labour intensive, and consequently they often involve a limited number of participants or amount of data per participant.

Several prior studies have observed similar limitations in the analysis methods used to understand other important areas of computer use, and this has motivated the development and deployment of logging tools to record data describing users' actual activities as they occur. Log-based analyses have been used to examine Unix command use [85], web navigation activities [174], window switching behaviour [172], and navigation within electronic documents [5].

However, although one previous log study examined the flow of file resources between applications [102], there are no prior publicly available log studies specifically describing file retrieval. Reasons for this are suggested by Bergman et al. [29]: first, there are privacy concerns in monitoring file use that can act as a disincentive for participation as well as a barrier for human-ethics approval processes; and second, developing robust logging software is a non-trivial software engineering exercise that, when done poorly, results in unacceptable instability in the users' computing environment.

The previous chapter described the implementation of FileMonitor, a tool to monitor file retrieval behaviour on OS X. This chapter describes the results of a four week study in which FileMonitor was deployed on the personal computers of 26 participants. Interviews with the participants following the log study helped to cross-validate the findings derived from the log analysis and clarify any unexpected or anomalous observations.

Results provide a rich characterisation of actual file use, presented in three sections focusing on each of the following: (1) *retrieved files* – describing the types of files used, their locations, frequency of access, etc.; (2) *retrieval methods* – characterising the tools used to retrieve files, such as search, file browsers, recent items tools, etc.; and (3) *file organisation* – briefly characterising use of organisation features such as moving and renaming files. The chapter finishes by discussing implications for design of next generation tools

for retrieving files, such as file browsers.

The characterisation is focused on retrievals of discrete files that are individually named and explicitly retrieved by users (such as word processing documents, spreadsheets, etc.), rather than files that are accessed implicitly through specific applications (such as email messages selected within a mail client). Applications are themselves files, and discussion is included that characterises their retrieval, although they are not a primary focus.

5.1 Background

An overview of file retrieval techniques is described in detail in Chapter 2. While there is extensive similarity in methods available between platforms, those offered on OS X are briefly summarised below to aid in interpreting the results.

The Finder is the navigational file browser offered on OS X, equivalent to File Explorer in Windows. It allows users to traverse through their file hierarchy to reach target files, and provides shortcut buttons for common navigation actions such as traversing "Back" to previously visited locations. The Finder also features a sidebar with customisable links to common locations, and a "Go" menu facilitating access to common locations and recently viewed folders. Naturally, the efficiency of file retrieval by navigating through the file hierarchy is influenced by the structure and clarity of the user-created file hierarchy.

The OS X Dock contains a single row of icons, most of which represent applications. The Dock serves a dual function, as both a method of switching between applications (all launched applications appear in the Dock) as well as a way to launch them. It can also store shortcuts to common folders and provides features to quickly access items within them.

OS X also provides a powerful search utility, called Spotlight. A Spotlight menu is available from within any application (including the Finder) and provides a listing of system-wide search results in response to a query, grouped by type (example types include documents, applications, folders, email messages, events, webpages, images and dictionary definitions). Spotlight searches all file metadata, such as filename, content, keywords, authors, and many other properties. A large number of search criteria and boolean

searches are available by using special keywords, however these are mostly undocumented and are not obvious to the user; for example, searching for "kind:folder date:today" shows all folders that were opened today. As of Mac OS X 10.7, Spotlight results show previews when hovering over a result, but do not show item locations.

The Finder also includes a search feature, powered by Spotlight. Once an initial search query has been entered, it provides an explicit user interface to filter the results based on many criteria – most obviously the search location and whether to search all content or just filenames, but also based on any other metadata. This interface can be shown via the Spotlight menu, by selecting a "Show All in Finder" menu item from its list of results.

Two additional methods are available to open files from within most applications: an open dialog, which provides a navigation interface similar to the Finder, and an "Open Recent" menu, listing the most recent documents opened in that application. A global "Recent Items" menu is also available as a submenu of the Apple menu, and lists recent applications and documents across the whole system.

Chapter 3 included a review of previous file retrieval and organisation studies. To summarise, key findings are that users average 6000-8000 thousand documents, but with large variation between users [82, 95], that there are strong preferences towards navigation-based retrieval [25], and that retrieving a file using navigation takes an average of 13-17 seconds [30]. However, none of these studies have directly observed retrieval behaviour in a natural setting over a prolonged time period.

5.2 Study Method

26 Mac users (11 female, mean age 31.9) participated in the study: five undergraduate students, twelve graduate students, five university staff members, and four employed outside universities. 14 studied computer science and 12 studied or worked outside computer science. Three reported that they used computers for 1-2 hours each day, four for 2-4 hours, nine for 4-8 hours, and ten for more than 8 hours each day. FileMonitor was installed on participants' primary personal computers; 8 were desktops and 18 were

laptops, split evenly between Mac OS X 10.6 and 10.7. Participants were offered a shopping voucher as thanks for participation.

At the start of the study, participants provided demographic information and completed an informed consent form (reproduced in Appendix A). To address privacy concerns related to the logging of file paths and filenames, they were assured that log files would be processed only by a computer program that produced aggregate data, and that no one would directly observe the contents of their logs. Logs were stored on participants' computers until collected, and participants could view their contents. This approach was preferable to concealing filenames (for example, using hash codes), as it allowed for analysis of filename information, and no participants declined to participate in the study after learning the details of what would be logged.

FileMonitor was installed for about a month, with four weeks of data being included in the analysis for each participant, starting on the day after installation; logged events that occurred on the day FileMonitor was installed were ignored as participants may have been more conscious that they were being observed. After the four week period, individual arrangements were made to uninstall the software, collect data, and conduct post-study interviews.

The interview was semi-structured and conducted along with participants' computers so that they could demonstrate their retrieval behaviours in response to questions. Sample questions included asking the participants to describe three files they had been working with over the last week, to demonstrate the ways in which they accessed them, to recall any file retrieval problems they had encountered recently, and to describe how they resolved any such difficulties. Participants were also probed on their usage of a wide range of file retrieval tools and methods. Questions were designed to encourage participants to offer information, rather than prompting for it directly. Interviews usually took about 30 minutes. The interview template is reproduced in Appendix A.

5.2.1 Limitations of Log Analysis

Logs were analysed by computer program to produce aggregate results. Due to limitations of FileMonitor, some assumptions were made while processing the results.

The most important limitations related to the way recent items were recorded. Recall that the system-wide recent items document was monitored to detect file retrievals that were not triggered by use of the Finder or Spotlight search. For such retrievals, these limitations were: (1) the 'recent items' file is not updated immediately, so the logged timestamps can be slightly delayed – this delay is normally less than a couple of minutes, but means that the order of retrievals can be unclear, and time-based retrieval measurements cannot be performed; (2) it is impossible to determine the exact retrieval method used, except that it was not navigation in the Finder or search (possibilities include recent items menus, open dialogs, and the Dock, among others); (3) it is possible, though unlikely, that some retrievals will not be observed – for example, if more files are retrieved between updates to the recent items file than the length of its access history; and (4) some applications trigger updates to this file when an already open document is brought into focus, overestimating the number of retrievals. This behaviour is not universal across all applications.

Two assumptions were used when processing the logs to reduce the impact of these limitations. First, that a non-application file was opened by dragging it from a Finder window to an application icon in the Dock if (1) it was recorded as opening while visible in a Finder window used in the last two minutes; and (2) that window has not recorded a retrieval since its last navigation event. In such cases, the retrieval was marked as one triggered by navigation within the Finder (limitation 2). Such retrievals are excluded from time based measures, since the timestamps are not accurate. Second, retrievals from this method for any non-application file within six hours of a prior retrieval of the same file were ignored if they were not coupled with an associated *close file* event (limitation 4). *Close file* events were only sent in limited circumstances: on Mac OS X 10.7 only, and only for files opened from the Finder where the file was still visible in the window from which it

was opened. As an example, if retrievals were recorded for a single file with this method at hours 1, 2, 4, 7, 14, 18, with a *close file* event at hour 16, then only the retrievals at hours 1, 14 and 18 would be included. Due to the limited accuracy of this approach, including both false positives and false negatives, only limited analysis was done using this data.

5.3 Analysis Part 1: Retrieved Files

The analysis results are reported in three parts. Part 1, reported here, examines characteristics of retrieved files such as their type, hierarchy depth, and revisitation patterns, as well as analyses of filenames. Part 2 discusses the tools used to retrieve files. While these two parts, which analyse file retrieval, are the focus of the work, Part 3 briefly discusses file management to provide additional context.

Across the 26 participants, 18473 retrieval events were recorded. The cross-participant range of retrieval events was from 27 (a casual computer user) to 3889, with a mean of 711 (s.d. 860). The 18473 retrieval events contained a total of 19288 files, with the discrepancy explained by single retrieval events sometimes opening multiple files (e.g., multiple selections in the Finder). The total number of unique files accessed was 8218, ranging from 20 to 1251 per participant (mean 316, s.d. 314).

5.3.1 How often are files revisited?

Figure 5.1 shows a breakdown of files by retrieval count, showing that 60.8% of files were only retrieved once (accounting for 26.6% of all retrievals), and 98.0% were retrieved 10 or fewer times. Half the participants accessed at least one file more than 20 times, and the maximum number of revisits to a file was 280 times.

Prior log analyses in other domains have shown strong patterns of revisitation, similar to that exhibited by Zipf's Law [189], with retrieval counts to distinct items following a power-law with their rank. These findings have been replicated for command use [85], web page visits [174], window use [172] and email messages [64]. It was therefore useful to investigate whether file revisitations follow a similar distribution; such knowledge is beneficial, for

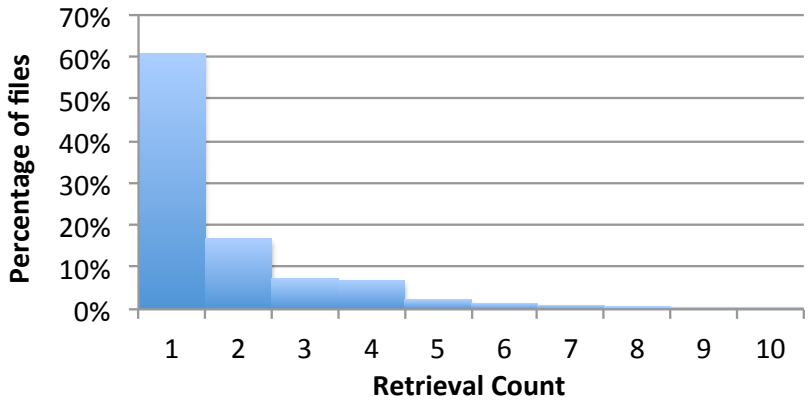


Figure 5.1: Distribution of files' retrieval counts, across all participants

example, when simulating file retrievals in controlled lab studies.

This was tested by using the regression $\log(r) = a - b\log(n)$, where n is retrieval count, r is the file's rank by retrieval count, and a and b are constants where the data is Zipfian when the Zipf's parameter $s \approx 1$ (where $s = \frac{1}{b}$). The analysis included the top 25 most frequently retrieved files of the 17 participants who retrieved more than 100 different files. Results showed good coefficients of determination (mean R^2 of 0.93, with individual's R^2 values ranging from 0.81-0.98), with s averaging 0.62 (range: 0.32-0.97), indicating a near Zipfian distribution. Restricting analysis to just those participants who accessed more than 500 files (6 participants) gave an even better fit: mean $s = 0.68$, mean $R^2 = 0.952$.

Therefore, people's patterns of file retrieval are strongly repetitive, with a small number of frequently revisited files, and a large number of infrequently visited ones. They are also approximately Zipfian, albeit with a longer tail than typical and with large variation between participants. The obvious design implication is that there are promising opportunities for interface methods that improve performance in revisiting files.

There is also evidence that users work in related areas of the hierarchy over limited time periods. For example, while 37.6% of retrieved files had been visited in the previous 24 hour period (51.8% in the previous week), 83.0% of their parent folders contained files (at any depth) that had been (90.1% for the previous week). Going further, 91.4% of their grandparent

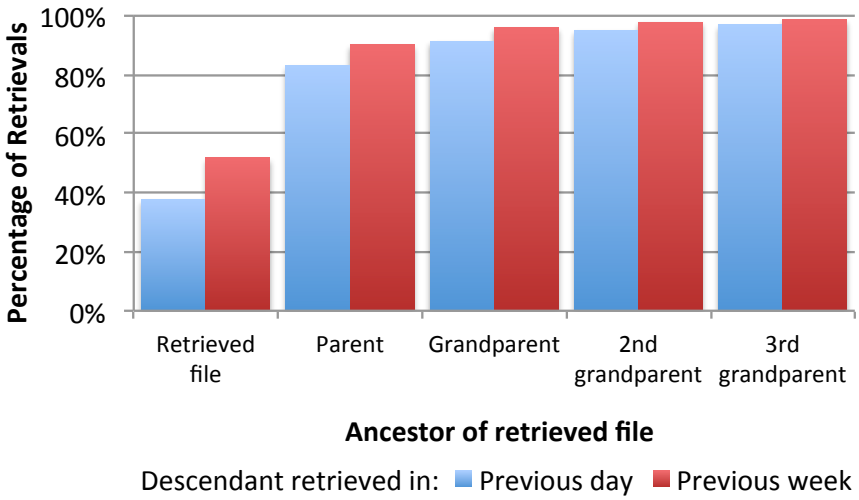


Figure 5.2: Percentage of ancestors of retrieved files that contained descendant files that were retrieved in the last day or week (including the retrieved file itself), as an indicator of reuse in parts of the file hierarchy.

folders contained files that had been (95.9% for the previous week). These figures, as well as those for further ancestors are shown in Figure 5.2.

5.3.2 Are the same items often accessed in each folder?

While the previous section shows that file revisitation is common, a relevant question for the design of potential file retrieval interfaces is this: given a particular folder, what is the distribution of retrievals amongst its items? Do a few items represent the majority of retrievals within a folder, or is there a more even distribution? Interfaces that support revisitation may act globally (for example, OS X's system-wide 'Recent Items' menu), however others may be specific to a folder (for example, an interface that highlights likely items in a folder). The latter type is most effective when revisitations are skewed to a small number of items in each folder.

This was examined by recording the frequency rank of each item within its parent folder at the time it is accessed, treating files and folders independently. For example, suppose a folder called *Documents* contains subfolders *Finance*, *Coursework* and *Projects*. *Projects* is accessed at one point of time,

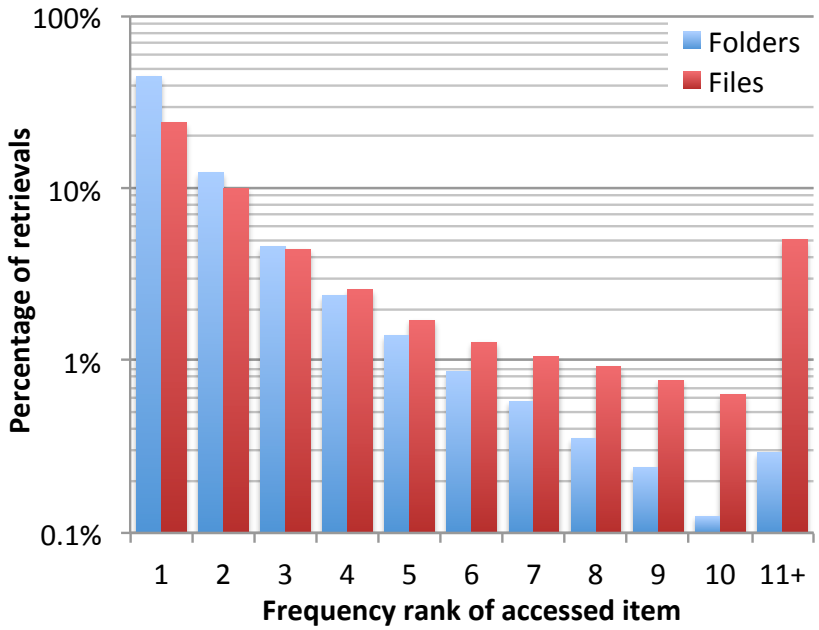


Figure 5.3: Frequency rank of files and subfolders within their parent folders, at the time they are accessed. Y-axis uses a log scale.

when only *Coursework* has been accessed more often in the past – thus, the folder with frequency rank 2 was accessed. Later, *Projects* is accessed again, but this time it has been accessed more than any other folder in the past, and has frequency rank 1. Figure 5.3 shows the distribution of such frequency ranks for folders and files across all participants. The first access of each type (file or subfolder) is excluded for each folder, since there is no access history. The first access of a particular item, where other items in that folder of the same type have previously been accessed, are included in the total percentages but not shown.

Notably, the frequency distribution for files has a much longer tail than for folders. Files and folders with frequency rank 1 account for 24% of file retrievals, but 44% of folder retrievals. At the other extreme, 5.0% of file retrievals are for files with frequency ranks of 11 or more, compared to just 0.3% of folder retrievals. Additionally, in folders with previous access history, 47% of file retrievals are for previously unvisited files, while only 32% of folder retrievals are for previously unvisited folders. These results are perhaps to

be expected, given that folders generally contain more files than subfolders [29].

These findings suggest that usage data is a useful predictor for future accesses within folders, and especially so for subfolders. Over 60% of folder retrievals (and 90% of folder revisitations) are for a folder with a frequency rank of 3 or better. Almost 40% of file retrievals (73% of file revisitations) are for files with rank 3 or better. For comparison, previous studies have found an average of 11-13 files per folder [29, 82, 95], while Bergman et al. found an average of 10.6 subfolders in the folders most commonly visited [29]. While this metric essentially tested the predictive accuracy of the *Most Frequently Used* method (MFU), other algorithms have been shown to be more accurate still (see Chapter 6).

5.3.3 What types of files are retrieved?

File types were deduced based on file extensions and grouped into categories. Types were counted for every retrieval of each item, however each type was counted a maximum of once per retrieval event, in the case of multiple selected items. Using this method, 91.1% of file retrievals could be classified. Common types of retrieved files were: applications (24.1%, including those opened indirectly by opening one of its documents), images (13.3%), text or word processing documents (12.7%), PDFs (12.5%), movies (12.2%) and source code (8.5%). All other types were less than 2% each. These results differed considerably from those found by an earlier study by Gonçalves and Jorge [82]. One explanation is the difference between the distribution of types amongst retrieved files (as in this study) compared to all files (as in Gonçalves and Jorge's study) – for example, applications are likely to have higher revisitation rates than other file types. It may also represent changing trends in computing since their analysis, with multimedia files now likely more common.

There were large differences between retrieval methods for different types of files. With the Finder, text documents accounted for 26.2% of file retrievals, movies 20.4%, PDFs 14.8% and images 9.0%. With Spotlight, applications accounted for 70.5% of retrievals, PDFs 10.4% and text documents

3.0%. With Finder search, PDFs accounted for 40.8%, source code 17.4% and text documents 14.3%. For *Quick Look*, image retrievals accounted for 33.9%, movies 19.3%, PDFs 16.2% and text documents 9.0%.

In summary, the predominant method for retrieving media files was navigation through the hierarchy using the Finder, and navigation-based retrieval with *Quick Look* was often used for images. Spotlight was used mostly to launch applications, and Finder searches were most often for PDF documents or other non-media documents. Possible reasons for these differences include: (1) media files are difficult to search for, as they do not have text-based content, so navigation is preferred; (2) when images only need to be viewed, Quick Look is often ideal since it does not require a dedicated image viewer to be launched and the full image can be easily viewed in a preview. Additionally, it provides an ideal way to quickly scan through multiple image files, either to browse a number of images, or to search for a specific one; (3) applications are well suited to Spotlight searches because the application name provides an easy-to-recall search query that has low cognitive load, and because application results are given high prominence in the search results; and (4) PDFs and other documents are suited to Finder search when search is the preferred method, as more advanced search criteria are often required than are provided in the Spotlight menu in order to produce a small list of candidate results, such as location restrictions.

Application retrieval behaviour was considerably different to that of other files, as should be expected given the various shortcut methods to retrieve them. All participants either used the Dock (most participants), Spotlight, or third party applications as their primary method of launching applications. Navigation was rarely used for accessing applications.

While some of these results are specific to the retrieval tool implementations on OS X, this important interaction between retrieval method and file type emphasises the need to consider the use cases of a new technique, and tailor its features to best support the types of files it is best suited to retrieve. The use of retrieval tools is more fully analysed in Part 2 of the analysis.

5.3.4 What are the characteristics of filenames?

Understanding how people name their files is useful for text based retrieval systems. For example, a keyword based system might need to accurately decompose the tokens of a filename, regardless of whether it is called 'My filename.txt', 'MyFilename.txt' or 'My_filename.txt'. Knowing what naming approaches are actively used is important for such systems.

Filenames of retrieved files averaged 18.8 characters in length (median 16, s.d. 12.67), including any file extension. Participant means ranged from 13.9 to 32.0, and averaged 19.7 (s.d. 4.6). These values were considerably larger than those found by Gonçalves and Jorge's 2003 study [82], who noted at the time that their lower value (mean 12.6 characters, s.d. 8.1) seemed to be a legacy from when filenames were limited to 8 characters on some systems.

Table 5.1 summarises statistics for the content of filenames, excluding any extensions. On average, 42.8% of participants' retrieved files contained numbers in their filenames, less than the 60% that Gonçalves and Jorge found. However, this varied considerably, with a standard deviation of 17.1% between participants, and a range from 21.2% to 72.5%. An average of 30.1%, 21.1%, 14.4% and 8.4% of retrieved files contained spaces, hyphens, underscores and periods, respectively. These figures also varied considerably between participants, and it was clear different people used different conventions for naming their files. While the majority of participants most often used spaces as word delimiters, four used hyphens most often, three used

Component	**Mean percentage**	**S.D.**	**Min**	**Max**
Numbers	42.8%	17.1%	21.2%	72.5%
Space	30.1%	14.9%	6.4%	58.9%
Hyphen	21.1%	14.6%	3.3%	53.3%
Underscore	14.4%	10.9%	2.5%	49.7%
Period	8.4%	6.8%	0%	24.8%
lower case	20.0%	13.5%	2.8%	52.2%
UPPER CASE	8.8%	10.6%	0.4%	36.4%
CamelCase	18.9%	10.0%	9.3%	45.4%
Other Mixed Case	51.3%	17.3%	28.3%	74.3%

Table 5.1: Types of content in filenames of retrieved files (minimum and maximum values exclude participants who retrieved fewer than 100 files).

underscores most often, and one used periods most often. There were also variations in the cases used in filenames. Mixed case (excluding camel case) was by far the most common (average 51.3%), but lower case (20.0%), upper case (8.8%) and camel case (18.9%) were still common for some participants; three participants had all upper case filenames for more than 25% of retrievals, as did ten with lower case filenames. On the other hand, 13 participants had fewer than 5% of retrievals with upper case names, and three had fewer than 5% with lower case names.

5.3.5 How deep in the hierarchy are retrieved files?

Two definitions of depth are used in line with Bergman et al. [29]. For comparisons between different methods, *structural depth* is the number of folders in the file's path, rooted at either the desktop (depth 0), home folder (depth 1) or root level of a disk (depth 1). For retrievals completed by traversing through the hierarchy using the Finder, *retrieval depth* is the number of steps traversed by the user to reach the folder containing the file, with a first step of opening a Finder window. For example, a retrieval of depth two might involve: (1) opening a new Finder browser window; and (2) following an alias to a folder that contains the target file. Entering a search query in the file browser is counted as a single step. The effect of navigation errors is filtered out using three approaches: (1) use of sidebar or toolbar links resets the depth to two (i.e., opening a Finder window and clicking a link); (2) revisiting a folder during a single retrieval resets the depth to that folder's original depth d; and (3) navigating to the child of such a folder while bypassing the folder itself (e.g. moving between children in column view) sets the depth to $d + 1$.

Retrieval depth can be further broken down into *absolute* and *incremental retrieval depth*. These are identical for the initial retrieval in a new Finder window. For reused Finder windows, however, a full traversal is not needed; the *incremental retrieval depth* is then the number of additional steps required to reach the next file, using the same navigation error corrections as above (but resetting folder depth history after each retrieval), and with window selection constituting an initial step. The *absolute retrieval depth*

is the retrieval depth of an additional file as if it was the first retrieval in its window. That is, any additional steps continue to increase the retrieval depth, but folder depth history is not reset after each retrieval. Both absolute and incremental retrieval depths are 0 for retrievals of items on the desktop. Table 5.2 provides an example of how the three depth measures work in practice.

Action	**Structural depth**	**Retrieval depth**	
		Absolute	**Incremental**
Open new window at home folder	1	1	1
Open *Documents* folder	2	2	2
Open alias to *~/Documents/Finances/2013/*	4	3	3
Open *Budgets* folder	5	4	4
Open file *Budget.xls*	**5**	**4**	**4**
Open file *Old Budget.xls*	**5**	**4**	**1**
Go back (to *2013*)	4	3	2
Open enclosing folder (*Finances*)	3	3	3
Open enclosing folder (*Documents*)	2	2	4
Go back (to *Finances*)	3	3	3
Open file *Trends.xls*	**3**	**3**	**3**
Open sidebar folder *~/Documents/Downloads/*	3	2	2
Open file *inflation.xls*	**3**	**2**	**2**

Table 5.2: Example of how different depth measures change as a result of navigation actions.

These three definitions of depth cover various use cases for such statistics. *Structural depth* is an analysis of hierarchy structure, *absolute retrieval depth* analyses the navigation effort to reach files under the assumption that a new window must be used for each retrieval, and *incremental retrieval depth* analyses the navigation effort actually performed in realistic settings (i.e., when windows can be reused).

Figure 5.4 summarises the average depths for retrieved items for all three types of depth. The most obvious finding is the large difference between absolute and incremental retrieval depths (means 3.1 and 1.3 respectively), especially considering they are the same for the first retrieval in each window. This suggests that navigation retrievals using existing windows is a common and efficient use case and that an exclusive focus on initial retrievals as in other studies (e.g., [29]) can be misleading. Window reuse is further explored in Part 2 of the analysis.

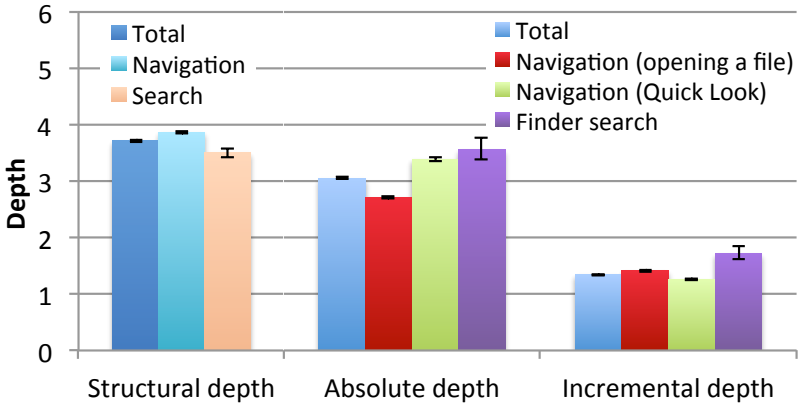


Figure 5.4: Average structural, absolute retrieval, and incremental retrieval depths. Error bars show standard error.

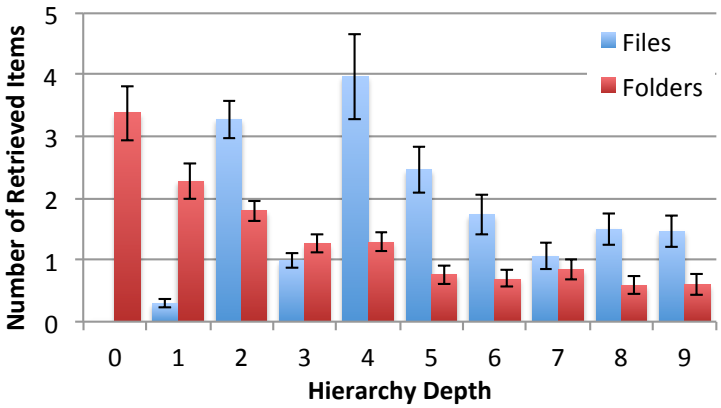


Figure 5.5: Average number of items accessed in folders at each level of the hierarchy. Error bars ± 1 st. err. for participant means.

Previous studies investigated the number of items at each level of the hierarchy tree [29, 82, 95]. Figure 5.5 shows a similar analysis, but where a tree was constructed from only the retrieved files and folders of each participant; files that were not retrieved, and folders that did not contain descendants that were retrieved, do not exist in the tree. Tree level 0 included the disks the participant used (mean 3.4 disks, s.d. 2.3 – note that this includes all mountable disks, including internal and external disks, USB sticks and network drives), and level 1 was the root level of these disks. The figure shows the average number of files and folders at each depth across all folders in the

retrieval trees.

There is a noticeable and consistent decreasing trend in the average number of subfolders retrieved per folder as depth increases, averaging 1-2 for depths 2-4, and less than 1 for higher depths. The number of retrieved files per folder, on the other hand, has a slightly less clear trend. Few files were accessed at depths 1 and 3, corresponding to the root level of disks and the depth of the home folder, both of which largely consist of subfolders. There were noticeable spikes one level deeper than each of these, however. At higher depths, they followed the same decreasing trend as folders. Overall, these numbers show that although folders average over 20 items each [29], most of these are rarely retrieved. Interfaces that take advantage of this fact to reduce visual search times could greatly reduce navigation times for common files.

5.3.6 Summary of Retrieved Files

Part 1 of the analysis examined characteristics of the files that participants retrieved during the study. Participants frequently revisited files, with revisitation patterns following a near-Zipfian distribution, and they typically accessed one of only a few items in each folder. There were large differences between the types of files retrieved with different retrieval methods. Participants used varying naming schemes for their files, with different preferences for word delimiters and cases, and average filename lengths have increased substantially in the last decade. Perhaps most importantly, a new measure of depth, *incremental retrieval depth*, showed that much less navigation effort is required to retrieve files in natural settings than traditional measures used in previous studies would indicate, as a result of significant window reuse. Part 2, below, further explores this, while also describing details about the ways that participants used each retrieval method.

5.4 Analysis Part 2: File Retrieval Methods

The analysis of files in Part 1 only examined quantitative data stemming from the log analysis. This section reviews the methods used to retrieve files, which is amenable to both log-based analysis and contextual interpretation

based on the subjective interview comments (e.g., why particular techniques were used and their perceived effectiveness).

Participants used a range of retrieval methods based on the type of the file, their memory of file attributes, and their current context. This section first gives a brief comparison of their use, before detailing the specifics of how each of navigation, search, recent items, open dialogs and other methods were used.

5.4.1 How does use compare between retrieval methods?

All of the participants retrieved files via navigation in a Finder window during the log study period and 85% used some form of search at least once, though only 46% used the Spotlight menu to search. 27% reported using the command line at some point, and 19% reported using a third-party file retrieval tool.

Tables 5.3 and 5.4 summarise methods participants used during the study period and reported using more generally.

Feature	Navigation	Spotlight	Finder search	Any search
# participants	26	12	20	22

Table 5.3: Retrieval methods used during the study period, based on logs

Feature	Recent items	Open	Dock	Cmd. line	3rd party
# participants	21	22	22	7	5

Table 5.4: Retrieval methods used by participants, based on interviews

On average, participants opened files in the Finder for 33% of retrievals (s.d. 20.8) and used *Quick Look* for 15% (s.d. 20.9), totalling 48% for navigation-based retrieval. Spotlight was used for 3.2% (s.d. 6.2) and Finder search for 1.0% (s.d. 1.0), totalling 4.2% for search. Other methods such as recent items, the Dock and open dialogs made up the remaining 48.2% (27.0% applications, 21.2% others), though these figures include applications launched implicitly by opening documents and assumptions about other document retrievals as previously outlined.

These other methods include specialised methods that are highly efficient for a small number of files (e.g. the Dock and recent items menus), as well as the navigation-based Open dialog. It is therefore clear that for files not available to these specialised methods, navigation is used the majority of the time (and likely a majority overall), while search is rarely used despite significant developments in recent years. These percentages are mostly consistent with user estimates in previous studies [25], although the actual search usage observed (4.2%) was considerably lower than the user estimates that Bergman et al. reported just for Mac users (13-15%), indicating that users likely overestimate their search usage.

How quickly do users retrieve files?

Non-controlled studies, such as this, necessarily have some ambiguity when performing quantitative measurements – in this case, in determining the start time for a file retrieval. The start of a retrieval in the Finder was determined as the later of (1) the time a browser window was opened; (2) the time of the first navigation event after either another retrieval, use of another browser window, or a file management task (e.g. moving or deleting an item); and (3) a navigation event that was more than 30 seconds after the previous one. For the Spotlight menu, it was defined as the first query change following a previous retrieval, any Finder use, clearing a query, or if the previous change was more than 30 seconds ago. The retrieval ended when the first item was retrieved. Retrieval times of any subsequent retrievals without intermediate navigation events or query changes were not considered. *Quick Look* retrievals were also excluded, as it was not possible to distinguish between its use to browse multiple items and as a substitute for opening files – both of which participants cited as reasons for using it. Though these times exclude any time before the first event is recorded (including the cognitive time involved in deciding which method to use, deciding a search query etc., as well as the time involved in physically triggering the initial event), ignore the effect of interruptions, and do not consider failed retrievals, they give a good overall estimate of retrieval times.

Using this analysis, the mean time to retrieve files using file browser

navigation was 10.2 s (s.d. 12.2, $n = 2367$). Spotlight searches averaged 5.7 s (s.d. 6.5, $n = 396$), including a mean of only 2.9 s for applications (s.d. 3.3, $n = 179$) and 8.0 s for other items (s.d 7.5, $n = 217$). Finder searches, however, took substantially longer, averaging 16.5 s (s.d. 13.4, $n = 45$). A possible explanation of this difference is that Finder search was often used for harder to find files, where more complex search criteria needed to be set up, or where it took longer to scan the results, whereas Spotlight was typically used to quickly access items when there was less ambiguity between results. Note that searching in the Spotlight menu, then using the "Show All in Finder" menu item to perform a Finder search for the same query, would be counted as a Finder search using the time of the "Show All" menu selection as the start time.

Navigation retrieval times can be further broken down by analysing individual steps. As with retrieval times, there was some ambiguity as to what constituted a step. Analysis of step times used a conservative definition to reduce the amount of noise: it included events within a single window that navigated to a deeper level of the hierarchy by opening a file or folder without the aid of additional features such as the Finder sidebar or history buttons. Steps were excluded if the user used search, performed a file management task or changed the window's view type, used another Finder window or if the step took longer than 30 seconds (likely indicating an external distraction or use of another application).

Figure 5.6 shows step times at different structural depths. There is little variation based on depth, except that very deep locations have lower average step times. There are three possible explanations for this: (1) users who have deep hierarchies may be faster at navigating through hierarchy steps, either through greater levels of expertise or due to deeper, narrower hierarchies reducing visual search time at each step; (2) that deeper locations contain fewer items [29], reducing visual search time; and (3) that deeper locations are more likely to involve the final selection of an item; Landauer and Nachbar [120] found lower step times for the last step of a retrieval in menu trees.

Investigating the first explanation, the effect indeed mostly disappeared when performing an analysis of just those participants who accessed files at depth 9 or more. The third explanation could be checked by performing a

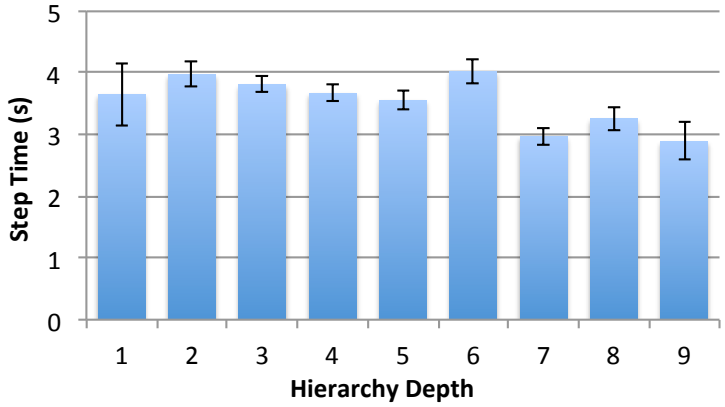


Figure 5.6: Average step times across all users at each level of the hierarchy. Error bars ± 1 standard error for global means.

separate analysis that examined step times based on the number of steps the user is away from the target file. This was assessed in hindsight after the target file was opened, and steps were numbered based on actual navigational steps rather than differences in structural depth. Steps that were part of extraneous intermediate navigation were ignored. These results, shown in Figure 5.7, actually showed *higher* step times for steps closer to the target file, and especially for the final step (i.e. opening the file), suggesting that the effect that Landauer and Nachbar found does not apply for file navigation. A possible explanation is that the number of subfolders in a folder is typically lower than the number of files [29]; if users know whether they are targeting a file or subfolder in a particular step and can quickly filter out items that do not match this criteria, then the set of potential target items is potentially larger for the last step when the target is a file. It is therefore likely that this contributed to higher step times at deep locations where file retrievals are more likely than subfolder selection, but that explanations 2 and 3 cancelled each other out to some degree.

5.4.2 Do users use different methods to retrieve the same files?

While participants reported that they each used a range of methods to retrieve files, a question remains as to whether particular files are always retrieved using a single method, or whether they are accessed using different

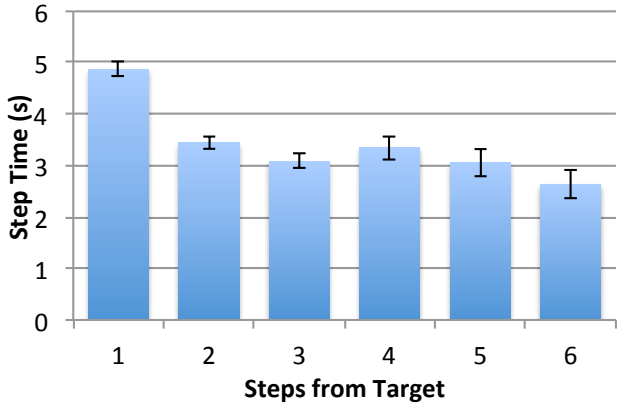


Figure 5.7: Average step times across all users based on the number of steps away from the target file (1 = target selection). Error bars ± 1 standard error for global means.

methods in different situations. Participant interviews indicated that factors such as the active application had some influence over the method used. The following analysis provides quantitative data about how often different methods were used for the same files.

Figure 5.8 shows interactions between the four methods that could be directly observed: use of the Finder to navigate to and open files (Finder navigation), to preview items (Quick Look), or to search for and open files (Finder search), as well as search within the Spotlight menu. The figure shows both the relative use of the four methods, as well as which methods are used to retrieve files following the use of a previous method for the same files. Figure 5.9 provides more detail for each of the four retrieval methods, showing the percentage of files retrieved with each technique that were previously or subsequently retrieved with other methods. Of particular note:

1. Files retrieved using navigation (either opened directly or accessed using Quick Look) were rarely also retrieved using search, but some were retrieved using other methods.

2. Similarly, files retrieved using Spotlight were rarely also retrieved using navigation, but were sometimes used using other methods – especially for applications. As search is primarily used as a method of last resort

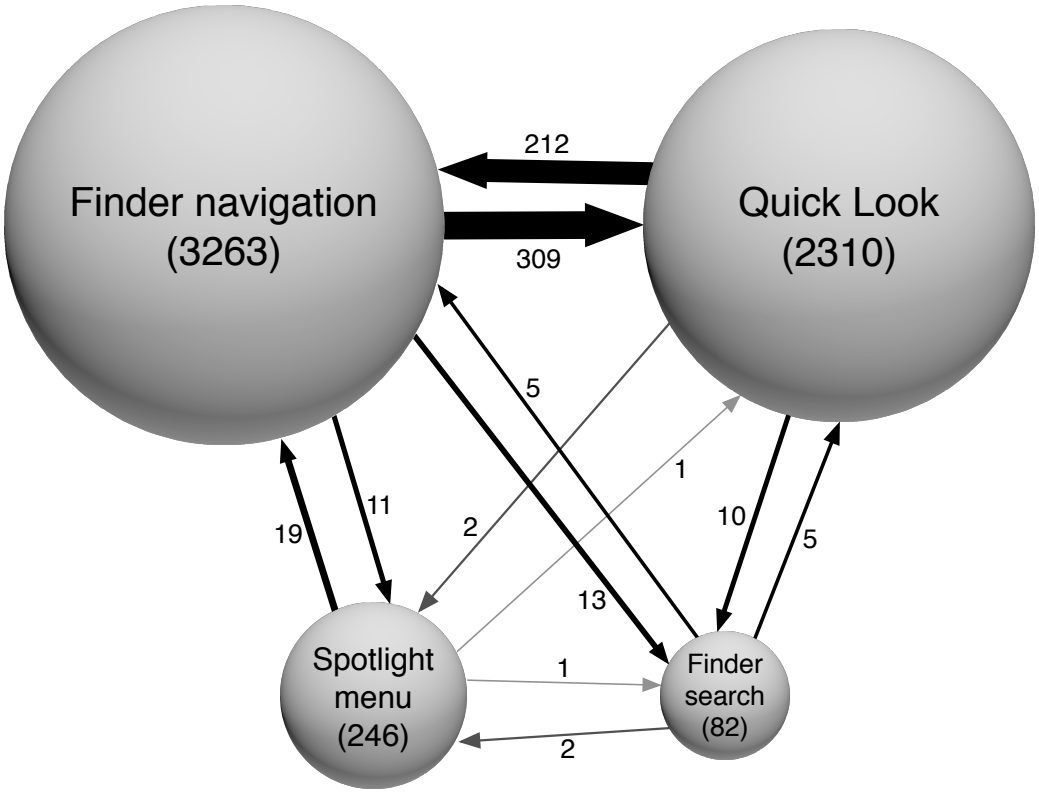
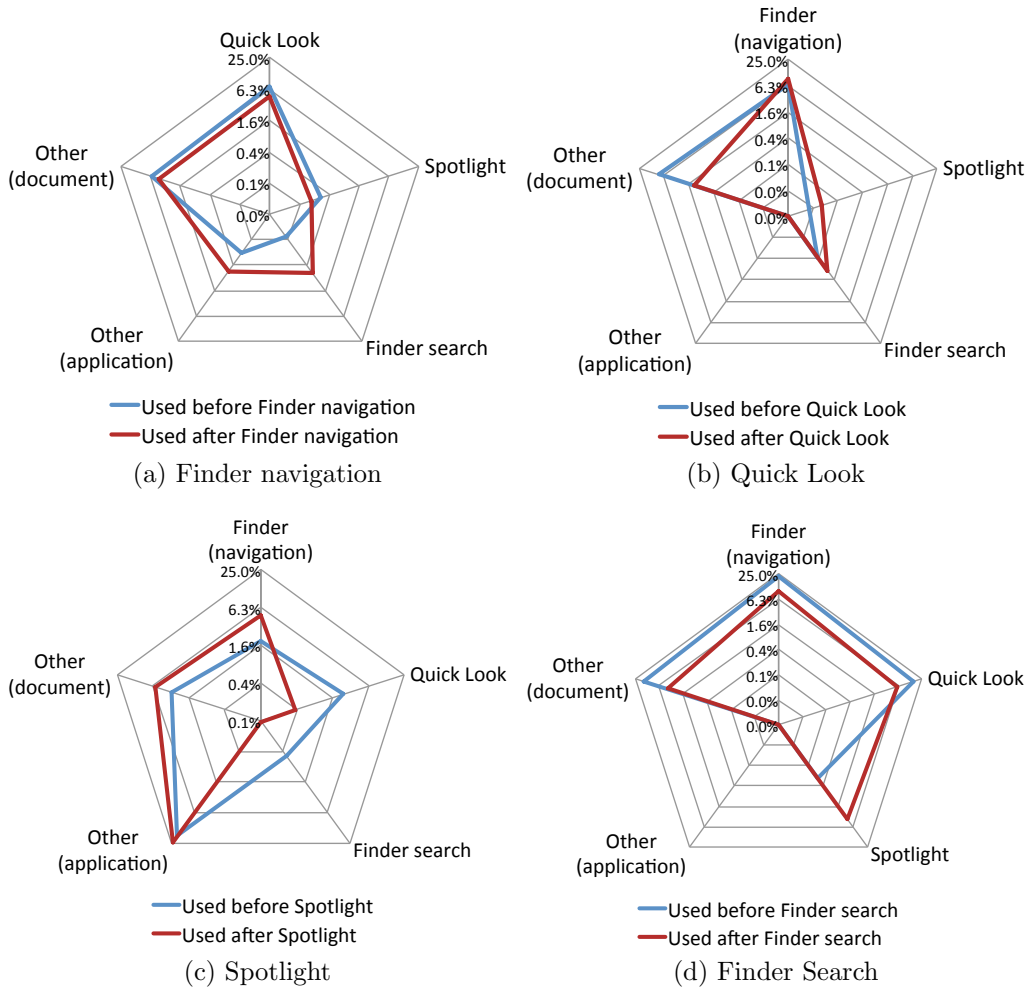


Figure 5.8: Retrieval methods and their relationships. Retrieval method numbers show number of retrieved files across all participants. Arrows show the number of files that were first retrieved with the source method, then later retrieved with the destination method. Sphere and line sizes shown to scale in three dimensions.

(see Section 5.4.4 for a discussion on the use of search), an implication of this is that when file locations are unknown, Spotlight does not facilitate location learning for later retrieval through navigation.

3. Quick Look and the Spotlight menu were rarely used for the same files. This is likely because Quick Look is commonly used for multimedia files, and rarely for applications – the opposite use cases to Spotlight search.

4. Navigation was more likely to be used for a file following an earlier retrieval of it in the Spotlight menu than vice versa, perhaps due to a



(a) Finder navigation

(b) Quick Look

(c) Spotlight

(d) Finder Search

Figure 5.9: Radar diagrams for various retrieval methods, showing the percentage of files retrieved with them (averaged across participants) that were first retrieved with each other method (blue lines) and later retrieved with each other method (red lines). Diagrams use log scale.

transition towards navigation for commonly access files. However, the opposite was true for Finder search: it was considerably more common to use Finder search to retrieve a file following an earlier retrieval using navigation than vice versa. This discrepancy can possibly be explained by the differing use cases of the two search interfaces, whereby Finder search is more often used to locate hard-to-find files whereas Spotlight search is more often used for quick retrievals (discussed later in this

chapter); files in the former category are likely to be rarely retrieved, perhaps resulting in little opportunity or incentive to rehearse navigation through the hierarchy to retrieve them. It is likely that transitioning from search to navigation is more common for more frequently retrieved files.

5. 17% of files were retrieved using multiple methods (grouping together all other methods such as open dialogs and recent items menus), or 44% of files which were retrieved multiple times. Grouping together the navigation methods (Finder navigation and Quick Look) and the search methods (Spotlight and Finder search), 11% of files were retrieved with multiple methods, or 28% of files which were retrieved multiple times. These relatively low figures suggest there is room for retrieval interfaces to better promote transitions to, and from, other methods where appropriate; for example, search interfaces could promote learning of file locations, and file browsers could directly incorporate support for revisitation to facilitate switching between navigation and specialised revisitation methods.

5.4.3 Navigation in the File Browser

All participants used navigation to retrieve files. In interviews, participants indicated that navigation provided improved context for their tasks, that recognition of file and folder names helped to locate items, that file management tasks could be performed at the same time as retrieval (e.g., copying a file prior to editing it), and that it was convenient to use when an existing file browser window was open and showing a folder in the same hierarchical vicinity as the target. However, not all comments were positive, with some participants stating that they found it tedious to use (*P2*: "I find a lot of navigation to be tedious... I just find it a pain to go through the hierarchy every time").

Several features of the Finder were either very lightly used, unknown, or unpopular. Only one participant used the "Recent Folders" submenu during the study period, and only used it a single time. However, another complained that there was no way to undo closing a browser window, perhaps

not realising that they could use "Recent Folders" as an alternative to this.

The "All My Files" feature was unpopular. This feature groups documents by type, and provides a horizontal scrollable list of files ordered by access date, with the most recently accessed files of each type initially visible. It became the default view when creating a new browser window in Mac OS X 10.7, but participants made only negative comments about it. *P6* complained that it just gives "a whole bunch of junk" and that he "[does not] really understand how it works", and *P10* said that "there doesn't seem to be any logic to it". These comments stress the importance of intuitive user interfaces; in this case, the feature was useless to many participants because they did not understand how it worked, could never foresee how to use it to reach any target file, and therefore did not use it.

The sections below describe how participants used various navigation features, including view types, the desktop and window reuse.

How are different views used?

The Finder provides four views: icon (a grid of icons), list (similar to the details view in File Explorer on Windows, with a row for each file and columns for various attributes), column (showing each new level of the hierarchy in a new list to the right of its parent) and Cover Flow (list view with a separate section showing file previews). Files can also be accessed from the desktop's icon view.

On average, participants used icon view for 12.3% of navigation retrievals, list view for 28.6%, column view for 42.8%, Cover Flow view for 5.3%, and the desktop for 11.1%. However, there was considerable variation in preferences. Preferred views based on usage (excluding desktop usage) were column view (12 participants), list view (9 participants), icon view (3 participants) and Cover Flow view (2 participants), with participants using their preferred view for 84.0% of retrievals (s.d. 17.2%). Most participants used column view (22 participants), list view (18) and icon view (17) for at least one retrieval, though only four ever used Cover Flow view (*P25* called it "useless"). In interviews, four participants specifically offered that they liked column view, though four others complained that narrow columns sometimes concealed

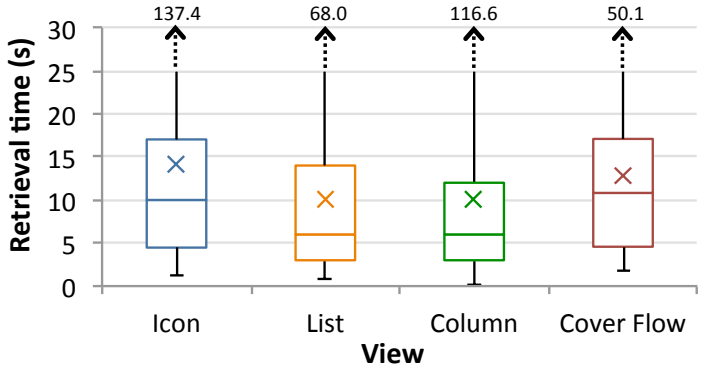


Figure 5.10: Retrieval times for Finder views. Whiskers show extremities. Crosses show means.

filenames. Overall, although participants occasionally changed views, they tended to have a strong preference for a particular one, with little consistency in preferences between participants.

Figure 5.10 shows retrieval times for the four view types. Mean retrieval times were lowest for list view (mean 10.0 s, s.d 10.3, $n = 554$) and column view (10.0 s, s.d. 11.9, $n = 765$), followed by Cover Flow view (12.9 s, s.d. 10.6, $n = 46$) and icon view (14.1 s, s.d. 15.9, $n = 203$). This difference can partially be explained by differences in incremental retrieval depths using each view; these were lowest for list view (2.3, s.d. 0.9, $n = 3712$), followed by column view (2.5, s.d. 1.1, $n = 5304$), icon view (2.9, s.d. 1.5, $n = 489$) and Cover Flow view (3.2, s.d. 1.2, $n = 96$). An analysis of step times showed that step times were lowest for column view (3.1 s, s.d. 3.7, $n = 3320$), followed by list view (4.4 s, s.d. 5.0, $n = 1183$), icon view (4.5 s, s.d. 4.5, $n = 622$) and Cover Flow view (5.3 s, s.d. 5.4, $n = 153$).

These results are perhaps somewhat intuitive. Cover Flow view would be expected to have slower step times than list view, since it provides the same view in a smaller area, requiring additional scrolling. Icon view requires visual search in two dimensions, and does not necessarily sort items, so may be expected to be slower than list and column views which are always sorted and only require scanning in one dimension.

However, the finding that column view was both the fastest per step and fastest equal overall, while icon view was the slowest of the three main views

per step and the slowest overall, is the opposite finding of previous research. Bergman et al. [30] found that icon view was significantly faster than other views on both Windows and Mac OS X. It is not clear whether the difference in incremental retrieval depths for each view were due to the types of retrieval tasks they were used for, or whether they better promoted efficient navigation (for example, column view facilitates navigation up the hierarchy, while a user of icon view might be more likely to restart a traversal from a sidebar link). The strong preferences that participants had for particular views limits the statistical power of such analysis, and neither this study nor Bergman et al.'s controlled for different types of retrieval tasks or folder structures. As such, further investigation is required to show which views are most efficient, and why.

What other navigation features are used?

The Finder includes a wide range of other features to assist navigation. Table 5.5 summarises how frequently these were used, showing that most are largely ignored. Only the sidebar was used by all participants, with most participants customising which folders appeared inside it. Of those that had done so, interview responses suggested that for most users these customisations were stable and seldom updated; a few participants, however, stated that they regularly updated the customisations to reflect changing needs. Six participants had also added items to the Finder toolbar, that logs showed

Feature	**Never**	**Seldom**	**Frequent**
Sidebar	0	4	22
Toolbar (for retrieval)	20	6	0
Back	7	8	11
Forward	21	5	0
Recent Folders	25	1	0
Go to Folder (type in path)	23	2	1
Go menu (specific folder)	17	7	0
Show Original (for aliases)	20	6	0
Enclosing Folder	22	2	2

Table 5.5: Use of navigation features in the Finder. Frequently used items were used more than 10 times in the study period.

were used during the study period. Navigation features available from the main Finder menubar such as shortcuts in the *Go* menu and the *Go to Folder* dialog that allows users to type a path were rarely used, perhaps because they have clear alternative methods (i.e., the sidebar and navigation). While the specific implementations of these features were specific to OS X, they translate well to features in other systems, and these findings indicate that menu items that aid navigation are rarely used, largely unknown and easily forgotten; instead it is preferable for navigation aids to be directly integrated into the main navigation interface.

How is the desktop used?

As reported above, on average the desktop accounted for 11.1% of navigation retrievals by participants (s.d. 20.2%). 20 of the 26 participants retrieved files from the desktop. Excluding one outlier (529 desktop retrievals), these participants averaged 14.4 desktop file retrievals each (s.d. 14.0). 11 participants opened disks from the desktop (mean 29.8 for these 11, s.d. 35.0) and 16 opened folders (mean 21.5, s.d. 38.6). Overall, the three activities occurred with similar frequencies; the main difference was in how many participants performed them.

Nearly all participants used the desktop for temporary storage, either for files that either were yet to be organised or would eventually be deleted. A smaller group stored items there for convenient access, such as for active projects. Some had screenshots automatically saved to the desktop. Several participants stated that they put items on the desktop with the intention that they would be there temporarily, but that the items would often remain for months or even years. Only one participant maintained a clean desktop with no items. These findings are consistent with previous work by Nardi et al. [145].

How are navigation windows reused?

Part 1 of the analysis found a large difference between incremental and absolute retrieval depths, indicating that window reuse is common. This section expands on this finding to further characterise window reuse.

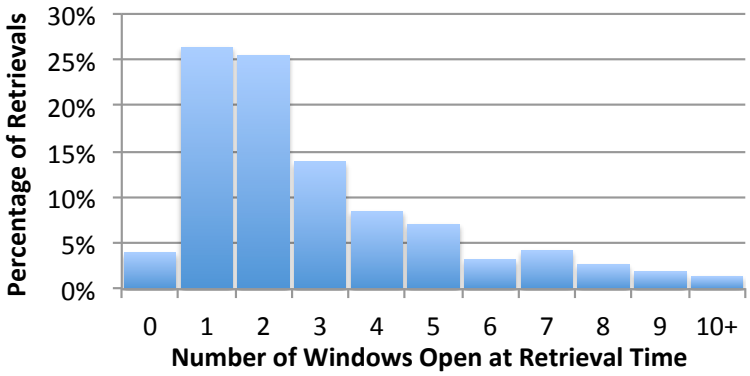


Figure 5.11: Number of browsers windows open at the time of navigation retrievals.

It is worth noting that as a navigational file manager, the Finder generally opens new folders in the same window, and that spatial file managers, which open new folders in new windows, likely exhibit different window usage behaviour. For example, they may average more windows open at once since more are created when traversing the hierarchy; on the other hand, they do not allow multiple windows to be open for the same folder. However, spatial file managers are becoming increasingly uncommon – current versions of all major operating systems use navigational file managers.

An analysis of window usage showed that participants had an average of 2.5 navigation windows open (s.d. 1.6) at the time of each navigation retrieval. There was a wide range in participant means; several participants averaged fewer than one window open at the time of retrieval (many retrievals were for items on the desktop when no windows were open), while others averaged close to six windows open at a time. Figure 5.11 shows the distribution of window counts at retrieval time, showing a long tail with many retrievals still performed with large numbers of open windows.

The next part of the analysis investigated whether multiple open windows led to significant reuse. In fact, 85.3% of navigation retrievals reused existing windows, with little variation based on the number of open windows except for a slight decrease with eight or more open windows (78.7%, covering 6% of retrievals). A possible explanation is that having a large number of open windows is an effect of being more likely to create a new window for a new

retrieval, rather than reuse an existing one. Alternatively, a large number of open windows may make it harder to find a relevant window to reuse.

Similarly, excluding windows which recorded no retrievals or were still open at the end of the study period, participants averaged 7.0 retrievals per window (s.d. 6.8). While most (17) participants had averages in the range of three to eight, five had much larger means of 11 to 31 (significant reuse), and five had much lower means which were close to one (infrequent reuse).

While reusing windows was common, it serves little benefit if it does not reduce navigation times compared to using a new window. Lower incremental retrieval times found previously show that window reuse was clearly beneficial. This is backed up by analysing how many path components (ancestor folders) consecutive retrievals in each window share in comparison to the same metric for all navigations. For this metric, the root of every disk, the home folder and the desktop were all treated as initial components. Within windows, average participant means were 3.5 folders in common between each pair of consecutive retrievals within the same window (s.d. 1.6). This compares to 2.8 folders in common for all navigation retrievals (s.d. 1.5), and 2.0 for retrievals performed with any method (s.d. 1.1).

The above results indicate that users gain considerable benefit from reusing windows, however a secondary question is then whether users reuse windows in the most efficient way, especially when they have lots of open windows. This was determined by calculating the percentage of the time that an optimal window was used when a window was reused; that is, how often, out of all open windows, they used a window that required the fewest navigation steps up or down the hierarchy to reach the target file (ignoring sidebar links, aliases, etc). Of all reuses where multiple windows were open, 94% of them used an optimal window. Surprisingly, this figure did not vary noticeably as the number of open windows increased; the optimal window usage remained between 93 and 96% when 8-10 windows were open, for example.

The average distance between consecutive retrievals in a window (i.e., the number of steps up or down the hierarchy required, including selection of the target item) was 1.48 (s.d. 1.23). The average *optimal* distance was 1.37 (s.d. 1.06), just 0.11 steps (s.d. 0.55) less. As these metrics ignore the possibility that participants deliberately avoided using an optimal window,

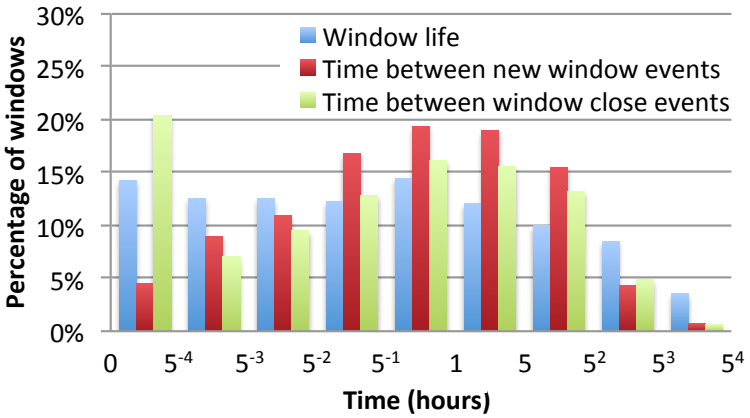


Figure 5.12: Window lifespans and times between new window and window close events, grouped into buckets and shown on a log scale. Total range is 0 to approximately 26 days.

which may have involved navigating away from a folder they would want to reuse later, this difference can be viewed as an upper bound. Combined with the high optimal window usage rate, this suggests that users reused windows efficiently even with a large number of windows open.

The final part of the analysis involved investigating how long users kept windows open, and whether they tended to close windows individually as they were done with them, or in bulk in a *clean-up* phase. Browser windows were open for a mean of 14.0 hours, but with an extremely large standard deviation of 46.4 hours and a large skew. In fact, the median time for a window to be open was just 10 minutes. Figure 5.12 shows the distribution of window lifespans, as well as inter-arrival times between user initiated new window and window close events. All three were fairly evenly distributed logarithmically. Of note, however, is the spike for window close events in the first time bucket; that is, 20.3% of window closes occurred within 5.76 seconds of the previous window close. This was much higher than the corresponding percentage for new windows (4.5%). This discrepancy can likely be explained by *clean-up* behaviour, with participants rapidly closing multiple windows in quick succession. Figure 5.13 shows the percentage of window closes in the first bucket (less than 5.76 seconds) by participant, indicating a wide range in behaviour; while a few participants never closed multiple windows in quick

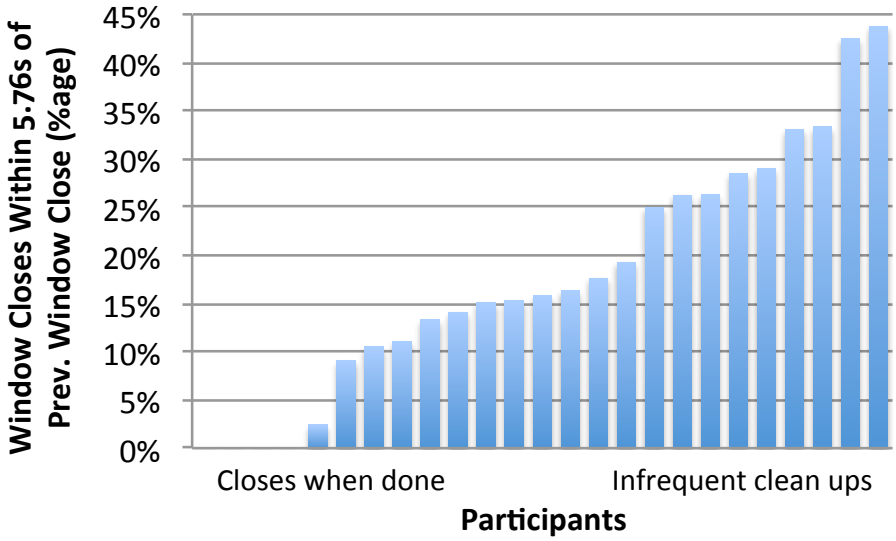


Figure 5.13: The percentage of occurrences of *window close* events in the (0, 5^{-4}) hour time bucket for each participant.

succession, others closed close to half shortly after another.

To summarise window reuse behaviour, participants often had multiple file browser windows open at once, reused them extensively, and did so efficiently. However, there were substantial differences between participants. Particular window usage behaviours were often linked together, leading to four clear categories of behaviour that participants could be grouped by:

Minimalists (5 participants) keep as few file browser windows open as possible and rarely reuse them. Most windows are not kept open for long. Minimalists tend to retrieve files less frequently.

Reusers (5 participants) keep file browser windows open and frequently reuse them – potentially for over 100 retrieval events in some windows. While they might create some temporary windows, most retrievals are done through the longer term ones. Reusers do not reuse windows as efficiently as other users, as signified by lower optimal window usage rates – target items often have little in common with the starting locations of the windows they are retrieved in. Reusers average two or three windows open at a time, preferring to reuse existing windows than create new ones.

Hoarders (3 participants) have large numbers of windows open at a time.

While they frequently reuse windows, they often create new ones rather than navigating away from existing window locations that might be useful in the future, or when a relevant window is difficult to find amongst the other windows. Once open, they are hesitant to close folders, but when they do they have a tendency to close multiple folders at once as part of a *clean-up* phase. Hoarders tend to retrieve a lot of files, though not all those that retrieve a lot of files are hoarders.

Optimisers (12 participants) sit between the extremes of the other categories. They use multiple windows, but not a large number of them. They reuse windows, but not excessively. When they do reuse windows, they do so more efficiently than those of the other categories, perhaps keeping a better mental model of the open windows and their locations.

5.4.4 Search

22 participants used search during the study, but for relatively few retrievals. Only one participant used it for more than 8% of their retrievals, and all used it less than navigation. This quantitative-based comparison matches the personal preferences expressed during the interviews: when comparing search to navigation, only two participants stated a preference for search; five stated that they preferred navigation to search, but would use search when navigation was likely to fail; and 16 indicated that they only used search as a tool of last resort, supporting prior findings [145, 25].

Why is search seldom used?

Participants gave many reasons for avoiding search unless necessary. Several mentioned the difficulty in differentiating document versions or items with similar names. This stems from the design of the particular search interfaces, which do not clearly show result locations; this is a common trait of many (though not all) search interfaces on other systems. Others mentioned the difficulty in using text queries to retrieve files with predominantly non-text content (such as multimedia files), automatically named files (such as images downloaded from a camera), or explicitly assigned file attributes (such as coloured labels).

A common complaint was that participants felt overwhelmed by search results, with too many results to be considered and insufficient ranking to assist rapid identification; in particular, they commented that there were too many obscure content matches or email messages (those who searched specifically for email would do so within their mail clients rather than with system-wide search). It was often difficult for participants to think of sufficiently specific queries to avoid an overwhelming results list, and most typically searched for filenames, possibly as an artefact of this. These issues dissuaded participants from using search; *P23* had "no confidence" in it and "[has not] been able to use it successfully", *P26* said "I'll use Spotlight, but I generally just find it frustrating", and *P1* concluded "I know [navigation's] not the fastest way... but I don't really care", echoing a common sentiment that reliability was more important than efficiency.

Several participants complained about the lack of discoverability of various features in the Spotlight menu. In particular, few realised they could open the parent folder of a result by holding down the *command* key while selecting it, but three mentioned that they desired such a feature, not being aware of it themselves. Another wanted to be able to perform advanced queries based on various metadata, but did not know the correct syntax. For searches within the Finder, some found it slow and found initial results for searches to be inaccurate, while one commented that he did not use it because it took him away from the current window location.

When is search used?

Despite these perceptions, participants recognised that search fills an important niche and offers many advantages in certain tasks. Participants indicated that they would use search when locations were unknown, for old documents, or when files were poorly organised. They also stated that search queries were normally based on filenames, especially when they thought the names were unique. Some of the participants stated that they were unconvinced of the practicality of content-based search: *P23* stated "I don't think of it that way. I think of the search as being for the file or the folder". Other uses for search that participants mentioned included generic searches about a topic without

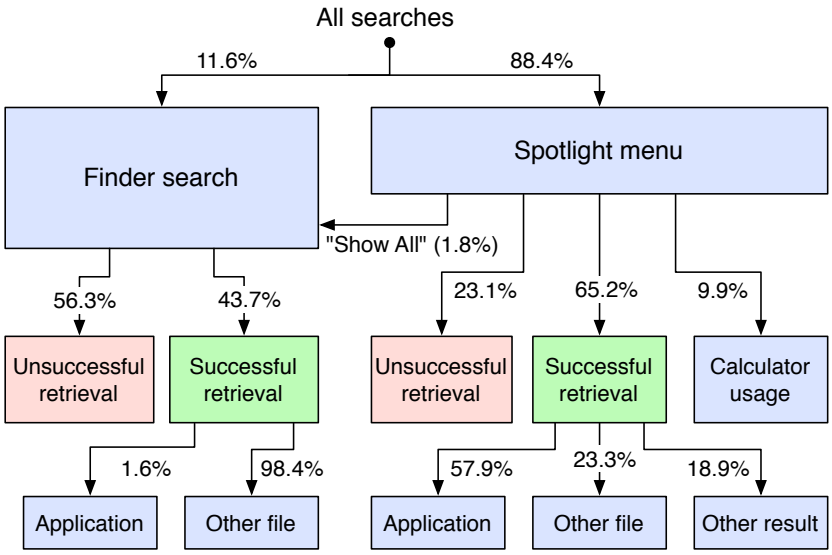


Figure 5.14: The combined search usage patterns of participants

a specific target in find, to find duplicates, to check if they had a file, or to search for a folder in order to create a browser window that they could reuse for multiple retrievals.

Six participants indicated that the Finder's search facilities provided a greater sense of control than Spotlight's. This was largely because the Finder's search provided an explicit interface to adjust search criteria based on various metadata, file management tasks could be performed on the search results, and that it was perceived as more easily accessible when transitioning to search after a failed navigation-based retrieval. Spotlight's most cited benefit was fast and predictable application launching.

How is search used?

Figure 5.14 shows the usage of the two search tools – Spotlight and search within the Finder – and whether they resulted in successful retrievals. The majority of searches (88.4%) used the Spotlight menu, although a small percentage of them (1.8%) transitioned to Finder searches by selecting the "Show All in Finder" item in the Spotlight menu.

Six participants extensively used Spotlight's calculator feature, where the result of an equation is provided as the top search result, accounting for

9.9% of total Spotlight searches. 73.8% of remaining searches resulted in retrievals; of these, 18.9% were non-file results (e.g. contacts, emails, system preferences or dictionary definitions), while most others were for applications. This confirms that many participants used Spotlight primarily as an application launcher, with search only a secondary function. Finder search, in contrast, was rarely used to retrieve applications, and was primarily used to search for other files. Logs indicated that 43.7% of these searches successfully retrieved a file, and 56.3% did not – a lower success rate than the Spotlight menu, but perhaps more indicative of the types of file searched for, rather than the search medium.

Despite Spotlight searching a range of file attributes, 89.7% of files retrieved with it had filename matches. Finder search allowed participants to specifically choose whether to search filenames or contents. Of these, 85.2% defaulted to, and remained, contents searches, while participants specifically changed it from a contents search to a filename search in another 13.4% of searches, with seven participants complaining about the default. The remaining searches defaulted to, and remained, as filename searches.

Finder search also provided an option for searching the whole computer (global) or just the current folder (local). Ten participants preferred local searches, compared to two who preferred global searches. Participants noted that their motivation for using Finder search instead of the Spotlight menu was often to search in a particular folder, though some changed the setting only if they could not find the target item in the initial search results across the whole computer. Indeed, 66.9% of Finder searches were global, and this was the default setting in 82.4% of searches; 19.7% of searches that were global by default were explicitly changed to local, while the reverse was the case in only 4.0% of local searches.

These findings have several implications. Participants rarely changed default search settings even when they knew the settings were there and wanted different settings, indicating the importance of sensible defaults. In this case, the result was less relevant search results, likely leading to lower success rates and slower retrievals. Defaults should also be tailored towards the likely use case of a particular interface – for example, even if most searches are intended to be system-wide, many of these searches are performed using the Spotlight

menu; when using Finder search, on the other hand, local searches are more likely and would make a more sensible default. Finally, while there has been much focus in recent years on content-based file search tools, participants overwhelmingly preferred filename searches for most search retrievals.

5.4.5 Recent Items

Of the *recent items* interfaces, the *File* ▶ *Open Recent* menu in document-based applications was most commonly used, though others such as the system-wide *Recent Items* menu, applications' Dock menus, and application-specific interfaces were used by some participants.

Several participants indicated that they liked the *Open Recent* menu's relative speed compared to other retrieval methods. Participants typically used it when the application was already active or following a restart or relaunch, though some participants would switch to the application specifically to use it. Participants mentioned several problems associated with the menu, such as its perceived lack of reliability when working with many files: *P15*: "I use a lot of different things, and by the time I get back to them they're not there anymore", *P13*: "most only show what's currently open". Other perceived problems included the difficulty in distinguishing between document versions, a lack of location context, and separate histories when working on multiple computers.

Only two participants used the system-wide *Recent Items* submenu, both for launching recent applications. About half did not know it existed, with several commenting enthusiastically that they would use it now they knew about it. A few participants mentioned that they used application-specific recent documents interfaces other than the *Open Recent* menu, and one mentioned accessing recent documents by selecting them from the application's Dock menu.

5.4.6 Open Dialogs

Nearly all participants used application *Open* dialogs at some point, mostly when already using an application or after failing to find a target item in the *Open Recent* menu. A stated reason for doing so over Finder-based nav-

igation was that it often provided quick access to the last used folder or to the application's default storage location. However, some participants noted a reluctance to open files this way, and used it only when other methods were particularly cumbersome or unavailable – for example, when the application was not the default for the target file's type, when importing, or for web applications. *P6* called the dialogs "completely obnoxious", with others complaining that it gave too small a view of the folder content. Two participants noted that they used open and save dialogs by dragging folders from the Finder to set the location, rather than navigating within the dialog, with *P6* calling this "completely backwards" in save dialogs, arguing that he should instead be able to drag a document to a Finder browser window to directly save it there.

5.4.7 Other Methods

Nearly all participants used the Dock to launch applications or to access folders such as *Applications* and *Downloads*. Seven participants infrequently used the command line to retrieve files, typically only for accessing hidden locations, or when already using the command line for other tasks. Third party retrieval tools were only used for application launching.

5.5 Analysis Part 3: File Management

While the focus of the study was on file retrieval, this section briefly discusses file organisation behaviour. Table 5.6 summarises the file management features participants used. Renaming, moving and deleting items were common, while creating new folders or giving items colour labels were less so. Excluding those who did not use a particular feature, participants renamed an average of 53.2 items, moved items an average of 27.8 times, deleted items an average of 74.6 times, created 9.6 new folders and labelled items 24.6 times.

File management events were most often performed in isolation, as shown in Figure 5.15. However, there was a tendency towards multiple such events occurring at once, with considerably higher probabilities in practice than what would theoretically occur given independent events, and given that file management events were about a third as common as the combined retrieval

Feature	Never	Seldom	Frequent
Rename file	7	9	10
Rename folder	6	8	12
Create new folder	15	7	4
Move file or folder	1	11	14
Delete file or folder	1	9	16
Set label	19	4	3

Table 5.6: Use of file management features in the Finder. Frequently used items were used more than 10 times in the study period.

events of the Finder and Spotlight menu. The (unsurprising) implication is that participants often performed a series of successive file management operations to reorganise their file hierarchy or delete files.

Those who put effort into maintaining their file structures rarely had problems retrieving files, though some noted their dislike for the time required for this maintenance. Others felt they had bad organisation, and large, unsorted downloads folders were common (*P6*: "I'm not encouraged to organise [downloads] with the current interface"). Downloaded files, screenshots and audio recordings often had unusual names and were hard to locate later if not renamed; two participants admitted that they often just downloaded files again if they could not find them (*P18*: "I do that all the time with lecture notes"). Others noted issues when they had multiple versions of documents, when they accidentally saved files in default locations without paying attention to where those locations were, or when they used shared repositories where others controlled the structure (*P24* referring to this as "really awkward").

5.6 Discussion

The results provide a characterisation of the files users retrieve and the methods used to do so. The main findings about the retrieved files relate to frequent file revisitation, with retrieval patterns that are approximately Zipfian, and that different types of files are accessed more frequently with different methods – for example, media files are typically opened using navigation, and PDF files are often accessed using search.

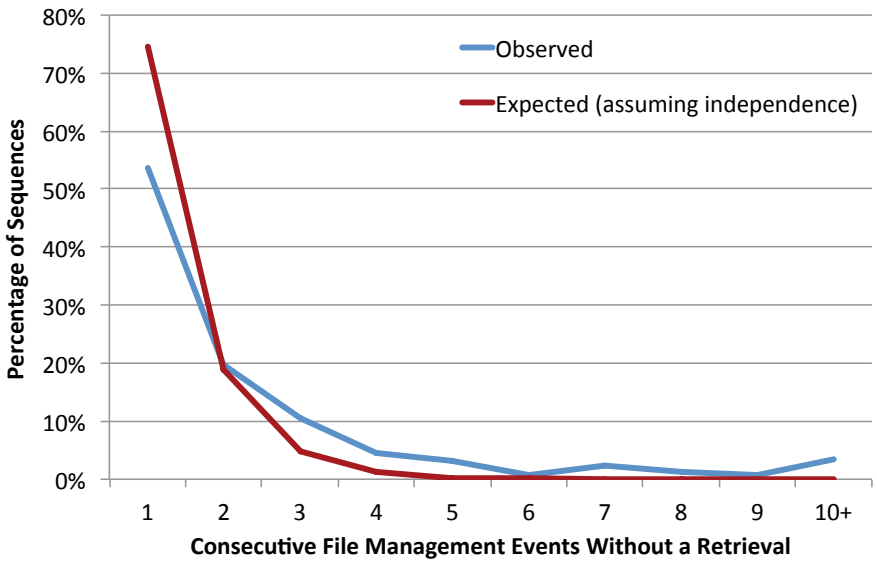


Figure 5.15: Consecutive file management events without a retrieval occurring, as a percentage of all file management sequences, averaged across participants. The blue line show the observed results from participants, while the red line shows the expected distribution if file management and retrieval events occur independently of each other.

The main findings on retrieval methods are that users have strong preferences for navigation-based file retrieval, and that users vary substantially in the ways that they use file browser windows. In particular, the analysis suggests that users broadly adhere to one of four characterisations of file browser window use: *minimalists*, who quickly dismiss windows; *reusers*, who maintain a small number of file browser windows; *hoarders*, who keep lots of windows open; and *optimisers*, who lie between the extremes. Spotlight's powerful search utilities were predominantly used as a quick method for launching applications.

Interview responses helped clarify why these patterns of behaviour were observed, as well as revealing perceptions of efficient and inefficient components of the interface (e.g., a strong dislike of the small view of folder content provided by the 'Open' file dialog).

The following sections compare the findings with those derived from prior studies that used other methodologies, and then discuss implications for design.

5.6.1 Comparison with Prior Work

Gonçalves and Jorge [82] previously analysed the proportion of files modified over various time periods by inspecting a static snapshot of file timestamp data. This provides a useful characterisation of the age of the files and the range of accesses through history, but it provides only a single data point for each file, thus hiding critical information about the frequency and temporal pattern of retrievals prior to the most recent one. Thus, the results in this chapter extend theirs with observations such as the near Zipfian distribution of accesses and the interaction between retrieval methods and file types. Additionally, the results show a substantially different distribution of file types compared to theirs, which is easily explained by changes and extensions to the range of purposes to which computers are applied in the decade since their study (e.g., media files are now more likely to form a large portion of retrievals) and the differences between the sets of all files and those actually retrieved.

Bergman et al. [30] found an average folder depth of 2.35 for Mac users, whereas the comparable figure for absolute retrieval depths was 3.1. The discrepancy is likely attributable to the difference in methods, with Bergman et al.'s study involving batched retrieval of a series of files contained in a snapshot of the recent documents list, whereas the study in this chapter involved real retrievals executed in pursuance of actual work over a month of interaction. The results related to *incremental retrieval depths* and a characterisation of window reuse go beyond those in Bergman et al.'s study, and would not have been possible without the dynamic data provided by longitudinal logs of behaviour.

The study also found lower retrieval times for navigation (10.2 s) compared to Bergman et al. [30] (12.6 s), however there were several differences in measurement; they started timing earlier (i.e., at the first mouse movement), included failed retrievals, and did not allow for window reuse. Future work will be needed to explain the discrepancy between step durations of different view types in the two studies. Nevertheless, the results confirm their findings that file retrieval is a time consuming task for such a frequent and important action.

The findings on search, namely that it was used by most participants as a last resort, echo several previous studies (e.g., [145, 25]). However, this study provides new insights into the relative benefits and usage of two different search interfaces (Spotlight and Finder search) which could help guide the design of future search tools.

5.6.2 Implications for Design

The primary aim of the log analysis presented in this chapter is to provide a broad characterisation of file use and file retrieval methods, and to provide a rich dataset that can be put to use by designers with varied intentions and needs. While it would be impossible to anticipate all the possible uses of the results, this section outlines broad recommendations for the design of file retrieval interfaces.

General Recommendations

Focus on worst case performance As illustrated by preferences towards navigation rather than search, users prefer reliability over efficiency, and will avoid interfaces that have a chance of failure or bad performance, even if they are faster on average. Interfaces should provide a convenient fallback in case of failure. Ideally, this fallback option should provide equivalent performance and cognitive load to alternative techniques available to them.

Provide sensible defaults Defaults should be tailored to the specific use cases of an interface manifestation. For example, even though most file searches may be intended to be system-wide, searches within the Finder are more often intended to be specific to the current window location. Interviews of search habits showed that users seldom change defaults even while complaining about them; 89.7% of Spotlight searches had filename matches, and 98.6% of Finder searches defaulted to content search, yet only 13.6% of these were changed to search only filenames even with the option easily accessible to users. Incorrect defaults therefore have the potential to have strong negative effects on both user experience and performance.

Avoid hidden features Features hidden in menus or otherwise out of view will generally be forgotten or never discovered. Useful features should be incorporated into the core interface, if this can be done in a simple, intuitive way. As an example, the Finder's "Recent Folders" submenu was only used by one participant, but several stated that such a feature would be useful. Features to facilitate switching between Finder locations, including recently closed locations, could be built into browser windows.

Recommendations for File Search Interfaces

Facilitate differentiation of items with similar names A limitation of search was that it was difficult for participants to distinguish between items with similar or identical names, an artefact of results being provided as a list, whereas in navigation distinguishing based on location is implicit. Search interfaces should provide a quick, intuitive way to distinguish between items based on location or other attributes.

Focus on attributes with high specificity Users are overwhelmed by large sets of irrelevant results when they are looking for one particular item. Searching a range of attributes, especially file content, leads to large result sets. Searches should instead focus on attributes with high specificity, such as filenames, which users are more likely to base their queries around; other results should be omitted unless specifically requested, or otherwise deprioritised.

Facilitate location learning Most participants indicated that they only use search as a last resort when other methods fail, yet navigation was rarely used after search for the same file, especially for harder to find items. This indicates a greater need for search interfaces to facilitate location learning to enable users to transition from search-based "last resort" interfaces to those that they prefer to use, such as navigation.

Recommendations for Navigation Interfaces

Aid revisitation, but only in context 90% of folder revisitations and 73% of file revisitations were amongst the top three most frequently visited folders or files inside their parent folder, despite previous research showing that folders average over 20 items each [29]. Interfaces that aid these revisitations could greatly improve performance. However, revisitation aids must be in context. Global revisitation interfaces, such as All My Files, were poorly understood and rarely used. Revisitation aids must be integrated into the core view, and provide a suitable fallback to ensure they are used (see *Focus on worst case performance* above).

Support window reuse Windows are often reused, greatly improving efficiency by eliminating the need to repetitively navigate to the same locations. File browsers should further embrace this paradigm, more heavily integrating features that facilitate location reuse.

Facilitate version control Participants often made their own backup copies of files, but found it difficult to manage multiple versions of a file. Features that facilitate differentiation between versions of files, as well as those that enable better ways to create or manage multiple file versions, could improve productivity and reduce the chance of lost data.

5.6.3 Implications for Evaluation

Retrieval interfaces are often evaluated in controlled lab studies, allowing for simpler evaluation methods and greater internal validity. However, it is important that such evaluations make correct assumptions in order for results to be generalisable to real world usage. The following provides guidance on several points based on the log analysis performed in this study.

File retrievals are near-Zipfian File retrievals can be approximated by a Zipfian distribution in lab studies that simulate file retrieval behaviour. A low s value should be used where possible, however participants vary considerably in their retrieval patterns, and a fixed s value should not necessarily be assumed.

Tasks do not happen in isolation Today's computers can handle countless simultaneous tasks, and multitasking by users is now the norm. Tasks often follow on from previous tasks, improving efficiency. Assuming that this is not the case, as in some previous studies (e.g., [30]), can lead to results that significantly differ from real world usage and miss important aspects of user behaviour.

5.7 Conclusion

File retrieval is an essential task for all computer users, and any improvements will result in considerable benefits. A thorough understanding of how people retrieve their files, and the reasons for doing so, aids the design of such improvements. The study described in this chapter is the first real world log study of file retrieval, and provides new insights into areas such as navigation window reuse and revisitation behaviour. Together with previous studies, the findings provide a detailed understanding of file retrieval behaviours that will be invaluable to the design of future retrieval interfaces.

Future studies will need to confirm that the findings generalise to other platforms, such as Microsoft Windows. However, the underlying retrieval methods are similar between platforms, and it is likely that insights based on the findings are generally applicable to all of them.

Part III

Predicting User Interaction

Chapter VI

AccessRank: Predicting What Users Will Do Next

Many forms of interaction with computer systems are repetitive – as well as frequently revisiting files, users also use the same commands [85], visit the same websites [174], return to previously visited document regions [5], and so on. To improve the efficiency of accessing previously used items, many diverse interactive techniques and systems have been developed, with examples including command histories [86], web page recency lists [110], scrollbar marks showing previous areas of document use [5], and menu adaptations that emphasise probable upcoming selections [67, 68, 11].

While there are plentiful examples of research and commercial systems that provide support for retrieving previously used data, there is much less previous research on the design and evaluation of the underlying algorithms that support the predictions presented to users. Improving the performance of these algorithms would have a strong impact on many areas of interaction. For example, Firefox's *AwesomeBar* uses the Places Frecency algorithm [143] to populate recommendations in a drop-down list alongside its URL bar, and membership of this list is progressively pruned and presented to users as they type characters, allowing predicted items to be rapidly selected.

This chapter introduces AccessRank, a new prediction algorithm designed for use in user interfaces. Rather than focusing purely on prediction accuracy, AccessRank is also designed to consider the stability of results over time – an important property in user interfaces, where spatial consistency can provide significant performance benefits.

The chapter begins by describing existing predictive algorithms and techniques from diverse areas of computer science that can be used to predict upcoming user actions. It then describes AccessRank and its calculation, combining two previous algorithms and adding new components to incorpo-

rate time of day information and to enhance stability of results over time.

In order to evaluate AccessRank, its performance was analysed across four domains: command use, web navigation, window switching and file retrievals. To achieve this, datasets for these domains were first converted to a standardised format. Next, a simulation compared the predictions of a range of AccessRank configurations and existing algorithms on these standardised datasets. Results of this simulation provide a detailed comparison of the accuracy and stability of these algorithms. Analysis also compares predictive properties of these domains and the effect these have on relative algorithm performance. The chapter concludes by discussing deployment of AccessRank in user interfaces.

6.1 Overview of Prediction Algorithms

Prediction is used in a variety of user interfaces, such as recent file interfaces, web page recommendations in an address field, and dynamic menus. The underlying prediction algorithms are often specialised for the particular domain they are used in, however many have common properties that can suggest useful elements of a domain-independent prediction algorithm. This section provides an overview of existing prediction algorithms in four contexts: menus, cache algorithms and two types of web prediction algorithms – those made locally in a user's web browser, and those that incorporate aggregate usage information from multiple users.

6.1.1 Menus

While menus normally present static lists of commands, many attempts have been made at incorporating dynamically ordered content. 'Open Recent' menus, for example, often include a list of recent files ordered by recency. These were more fully reviewed in Chapter 2.

Cockburn et al. [46] developed a mathematical model for predicting menu item selection time, taking into account Fitts' Law [70], the Hick-Hyman Law [101, 97] and user expertise. They noted the importance of stability in the context of human computer interaction. While efficiency for a computer might mean having items as close to the start of a list as possible, humans

who have gained expertise with a list are inclined to look for an item where they have previously found it. Sorting by frequency inherently provides more stability than doing so by recency, and consequently Cockburn et al. found that user performance was considerably better when sorting in this way. They also found that completely stable menus – those with static content – outperformed those ordered by recency, emphasising that low levels of stability can impede performance.

In another study examining relative performance of different techniques for ordering menu items, Findlater and McGrenere [67] compared three types of split menus. In all three, up to four items from the menu appeared in the top section, with the remainder in the bottom section and with a separator between the groups. They tested *static menus*, where the top four items were the four most frequently selected items, *adaptable menus*, which allowed users to customise which four items appeared in the top section and what their order was, and *adaptive menus*, which selected two frequent items and two recent items to place in the top section. They found that the frequency-based static menu performed better than the adaptive menu, with mixed results for the adaptable menu – consistent with Cockburn et al.'s finding that the inherent stability of frequency-based selections results in improved performance. Nevertheless, the algorithm used to generate the adaptive menu content can be generalised as *Split Recency and Frequency* (SR&F), with parameter n. SR&F selects the n most recently accessed items and orders them using *Most Recently Used*, then orders the remaining items using *Most Frequently Used*.

More complex algorithms have been incorporated into commercial software. For example, Microsoft Office 2000 implemented *Adaptive Menus* [11], which initially show only commonly used items but can be expanded to a full list by waiting several seconds or by selecting a menu item that explicitly expands the list. The underlying algorithm to select these items stores a usage count for each item, which is incremented each time the item is selected. To reduce the effect of older menu item activations, it also stores a last session count for each menu item, which is set to zero when the item is selected and incremented at the start of each new session. When the last session count reaches certain values (3, 6 and 9), the usage count is decremented by one.

At a higher value (12), the usage count is decremented by the larger of 2 or 25%, while for a higher value still the command usage record is deleted completely. Items with positive usage counts are shown in the shorter commonly used items list, while other items have their usage record deleted and are only part of the full list. The algorithm combines both frequency and recency; the usage count is a measure of frequency, while the last session count, a measure of recency, is used to impose an aging effect on the usage count.

6.1.2 Cache Algorithms

Cache algorithms are used to determine the most efficient set of items to maintain in a cache of a fixed size, in order to increase the probability that an accessed item can be quickly retrieved. These algorithms have the opposite goal of many prediction algorithms; instead of predicting the *most* likely item, they aim to predict the *least* likely, which is removed from a cache in order to make room for a new item. Despite these opposing goals, an algorithm that can estimate which item will be unused for the longest can often be adapted to estimate which items are likely to be accessed soon.

One family of cache algorithms is *Least Recently Used* (LRU) (e.g., [131]), which, as the name suggests, discard the items that are least recently used. In reality, approximations are often used due to the implementation costs of an exact implementation. In contrast to LRU, *Most Recently Used* (MRU) is another family of cache algorithms designed for use with cyclic access patterns [177], where older items are *more* likely to be accessed.

Least Frequently Used (LFU) algorithms remove items that are used the least. Generally they use LRU in the case of ties. Variations of LFU take into account recency by applying an aging policy that reduces the reference counts of items over time [159, 188]. Zhang et al. [188] identified several different categories of aging algorithms. *Explicit aging* involves periodic direct modification of item weights, and these algorithms effectively act as an add-on to an existing algorithm. *Value aging* algorithms, on the other hand, have aging factors built into the main weight calculations. Explicit aging can again be broken down into two subcategories. For *uniform aging*, the only

input is the original weight of each item and aging is applied in the same manner for all items. An example is α-aging, with uses the aging function $f(v) = \alpha v, 0 \leq \alpha \leq 1$. *Differential aging* also considers the distinct access patterns for each item, meaning that the relative ordering of items could potentially change when weights are modified.

Adaptive Replacement Cache (ARC) [137] takes account of both frequency and recency, resulting in better performance than LRU [138]. It uses two lists, one for recently accessed items and one for frequently accessed items, and also tracks *ghost* entries for each list. *Ghost* entries have had their actual data removed from the cache, but still contain the attached metadata. Hits on recently accessed items, regardless of whether they are ghost entries or not, result in the item being moved to the frequently accessed items list. While this algorithm cannot be directly adapted to predicting which item is most likely to be accessed next, it could be used to produce a set of items which is likely to contain subsequently accessed items.

Another algorithm that combines recency and frequency is *Least Recently/Frequently Used* (LRFU) [122]. It gives each item a *Combined Recency and Frequency* (CRF) value which is calculated based on the timing of all previous accesses. Parameters to the CRF function, shown in Equation 6.1, determine how much weighting is given to frequency versus recency, so it can be adapted for different situations. w_f is the item's weight, n is the number of past accesses, t is the current time and t_i is the time of access i (where time is in terms of discrete events).

$$w_f = \sum_{i=1}^{n} \frac{1}{p}^{\lambda(t-t_i)} \tag{6.1}$$

Typical metrics for comparing the performance of cache algorithms are *hit rate* and *byte hit rate*. The hit rate is the percentage of requests that are found in a cache, while the byte hit rate is the percentage of data found in a cache. These metrics are often inversely correlated [160] – strategies optimised to increase the byte hit rate are more likely to cache large items, reducing the hit rate as fewer items are stored in the cache overall. In the context of determining which items will most likely be accessed next, hit

rate is the most appropriate metric since item size is not a relevant variable. However, care must be taken interpreting results of performance studies for cache algorithms for application in other domains, since certain strategies give preference to small files, creating an artificially higher hit rate.

Several studies have compared the performance of cache algorithms. Romano and ElAarag [160] evaluated a large number of cache algorithms on several data sets. While most of the algorithms could not be adapted for predicting item accesses, there were still some relevant findings. They categorised algorithms into three groups: recency, frequency and recency/frequency. Within each group, there were large variations between algorithms. Comparing the results for the best algorithms in each group using the hit rate metric, frequency performed the worst, followed by recency, and algorithms using a combination performed the best. In another study, Arlitt et al. [12] found that frequency-based algorithms performed better than recency-based ones.

6.1.3 Web Browser Suggestions

Most web browsers feature some form of autocompletion when entering URLs in their address bars. In simpler implementations, they suggest URLs which are prefixed by the text entered by the user (excluding standard prefix strings such as 'http://' and 'www'). Modern browsers such as Firefox [144] and Safari [10] provide more advanced suggestions. Firefox's *AwesomeBar*, for example, suggests URLs based not only on prefixes, but matches throughout page titles and URLs of visited pages. It gives higher ranking to pages where the search term is the start of a token, for example in the URL *http://www.mozilla.org/firefox/*, some tokens would be 'mozilla', 'org' and 'firefox'. This feature helps to recall URLs when a keyword is not contained in the website's domain.

Unfortunately, a lot of queries typed into address fields will match a large number of results, and it is therefore important that they are suitably ranked. Firefox's AwesomeBar uses the Places Frecency algorithm [143], which ranks URLs by a *frecency score*, taking into account both recency and frequency. To calculate this frecency score, the algorithm first samples the most recent

ten visits to a URL. For each, it gives a weight from 10 to 100 based on how recent the visit was, then multiplies this weight by a percentage bonus based on the type of visit, such as whether it was typed in the address bar, clicked as a link on a webpage, or selected as a bookmark. The algorithm then takes the average of the weights for each sampled visit and multiplies the result by the total visit count to calculate a final frecency score. These frecency scores are quick to calculate and result in a reasonably accurate ranking. However, counterintuitively, unsampled visits (by definition, those older than the most recent ten visits of an item) increase the frecency score of an item more than sampled visits that are below the average sample weight. While designed for use in predicting URL visits, the Places Frecency algorithm can be adapted for general use by ignoring the visit type bonus.

6.1.4 Web Pages

Web page use can be analysed from either client-level history or server log data. The previous section focused on client-level predictions. This section examines server-level predictions, which can be used both for predictions about what a specific user will access next, such as for prefetching and recommendation (e.g., [99, 116]), and for predictions about what pages are most likely to be accessed by *any* user, which is useful for server optimisation. In both cases, aggregate data across all users can be used to improve prediction accuracy by analysing general access patterns. On the other hand, server-level data has several disadvantages: not every request will be logged due to client-side caching; servers generally identify users by IP addresses which do not necessary have a one-to-one mapping with users; server logs can be contaminated with data from non-human users such as web crawlers; and it can be difficult to obtain and analyse cross-website data.

A common approach to server-level analysis of log data is known as *web usage mining*. Cooley et al. [51] define web usage mining as "automatic discovery of user access patterns from Web servers". Gry and Haddad [78] evaluated several web usage mining approaches across several datasets to predict users' next requests, identifying the following three approaches:

Association rules [36, 73] – Given sets of items that are frequently visited together by users, association rules detect partial subsets that have been visited by a particular user, and infer that the unvisited items in that set are likely to be accessed. This approach ignores time as a factor.

Frequent Sequences [141] – These consider N-Grams in time-ordered sequences of URL visits by past users, and predict a URL k if the user's recent history exactly matches all but the last item k of an existing N-Gram. This was found to give the best accuracy of the three approaches when the correct prediction was required to have a high prediction rank.

Frequent Generalised Sequences [76, 129] – Similar to the *Frequent Sequences* approach, these also allow for wildcards in sequences rather than simple N-Grams. Wildcards can match one or more URLs.

These approaches share in common the property that they use aggregate history across multiple users to make predictions for a particular single user – a technique that is possible on the web, but difficult in many other domains where aggregate data is not available. However, techniques such as analysis of N-Grams are still possible in domains with a large access history and high revisitation rate.

Markov models can also be used to predict web page accesses [56]. These models assume that page accesses are *memoryless* – that is, the probability that a particular page will be accessed next is based only on the current web page and not on other browsing history. Such a model makes particular sense for web pages, since each page contains a set of links to other pages. Chances are high that the next page accessed will be one of these links, while the chance that the next page will be a link from a previously visited page is much lower. However, this assumption is less realistic when users have multiple tabs or windows open simultaneously, as server-side Markov models will not be able to detect which page the user is currently interacting with. Sarukkai also used Markov chains for link prediction [164], but recognised that noise like the above exists. To rectify this, he incorporated a user's link

history into his model by using weighting coefficients to give some influence to transition matrices of previous states.

Other research has examined clustering of accesses. For example, Perkowitz and Etzioni [150] calculate co-occurrence frequencies between pages to create a similarity matrix, which is used to semi-automatically improve the organisation of websites by learning users' access patterns. For each pair of pages p_1 and p_2, $P(p_1|p_2)$ is the probability that a user will visit p_1 if they have already visited p_2. The co-occurrence frequency is defined as $\min(P(p_1|p_2), P(p_2|p_1))$. Clustering techniques can be adapted for prediction, and are loosely related to *association rules*, described above.

6.1.5 Summary of Predictive Algorithms

This section introduced a range of prediction algorithms and techniques from several areas of computer science. In particular, the following algorithms have been described which are used as part of the evaluation later in this chapter:

- *Most Frequently Used* (MFU) and *Most Recently Used* (MRU – not to be confused with the family of cache algorithms of the same name). These are the most primitive prediction algorithms.

- *Split Recency and Frequency* (SR&F) [67], which selects n items with MRU, then the rest with MFU. The evaluated implementation uses $n = 5$.

- *Combined Recency and Frequency* (CRF) [122], originally used as part of a cache algorithm. The evaluated implementation uses parameters $(p, \lambda) = (2, 0.1)$, a configuration that performed well in initial testing on a subset of the datasets described in this chapter.

- The *Adaptive* algorithm [11], used to filter menus in software such as Microsoft Office 2000. The evaluated implementation orders item by usage count, using MRU to determine ties.

- The *Places Frecency* algorithm (PF) [143], used by Firefox's AwesomeBar to rank URL suggestions. The evaluated implementation excludes

features specific to web page prediction, enabling prediction in other domains.

- A *Markov* chain [132], adapted to make predictions using the following formula:

$$P(X_{n+1} = x | X_n = x_n) = \frac{|x_n \rightarrow x|}{|x_n|} \tag{6.2}$$

Here, $|x_n|$ is the number of previous occurrences of state x_n and $|x_n \rightarrow x|$ is the number of previous transitions from state x_n to x. X_i represents the state at time i. Given the most recent access x_n, the calculated probabilities provide a ranking, and MRU can be used to break ties. Variations that use higher order chains or that weight multiple chains together were investigated, but they had a negative effect on prediction accuracy in comparison to the above approach.

6.2 AccessRank

AccessRank's goals are twofold: first, to accurately predict the next action based on past ones; and second, to maximise list stability. The importance of prediction accuracy is obvious, but the need for stability is also important because it allows users to learn item locations over time, facilitating expertise with the interface used for list presentation [46].

AccessRank has two components, described below. First, a raw *AccessRank Score* is calculated, combining the *Combined Recency and Frequency* (CRF) cache algorithm (an algorithm with reasonable accuracy and stability), a *Markov* chain (a technique that produces high accuracy but low stability) and a *Time Weighting* (which weights items based on the current time and day). Second, a *Switching Threshold* improves results stability. Computing AccessRank is fast, as its model can be updated in $O(1)$ time, all scores can be updated in $O(n)$ time, and predictions made in $O(n \log n)$ time.

6.2.1 AccessRank Score

A raw AccessRank score w_n is calculated for each previously accessed item n using Equation 6.3. w_{m_n} is the Markov weight, w_{crf_n} is the CRF weight

(parameters $\{p, \lambda\} = \{2, 0.1\}$) and w_{t_n} is a time weighting. The AccessRank parameter $\lambda > 0$ determines the blend between the Markov and CRF algorithms, and can be adjusted based on whether accuracy or stability is more important or based on particular properties of the domain; Markov weights often provide better accuracy, at the expense of lower stability.

$$w_n = {w_{m_n}}^{\lambda} {w_{crf_n}}^{\frac{1}{\lambda}} w_{t_n} \tag{6.3}$$

The Markov weight is altered from Equation 6.2 to always give non-zero weights:

$$w_{m_n} = \frac{|x_n \rightarrow x| + 1}{|x_n| + 1} \tag{6.4}$$

6.2.2 Time Weighting

The time weighting w_{t_n} gives higher weighting to items that have historically been more frequently accessed at the current time of day or day of week. Informal observations indicate that many aspects of interaction are temporally predictable, for example habitually accessing a news webpage on arrival at work. Let c_h be the current hour of the day. For item n, let r_h be the ratio of the number of previous accesses of n in hours in the three hour range $[c_h - 1, c_h + 1]$, compared to the average number of previous accesses of n for a three hour slot. Similarly, let r_d be the ratio of the number of previous accesses of n on the current day of the week to the average across all days of the week. r_h and r_d are set to one if fewer than 10 accesses in total have occurred in the corresponding slot. The time weighting is then calculated as in equation 6.5:

$$w_{t_n} = max(0.8, min(1.25, r_h r_d))^{0.25} \tag{6.5}$$

6.2.3 Switching Threshold

The switching threshold improves stability. Consider items A and B at positions $r_A < r_B$ in the previous prediction list, where a lower position corresponds to a more confident prediction. Pairwise comparisons between item weights are made during sorting to generate a new prediction list. If A and B are compared and their new weights w_A and w_B are such that $w_B > w_A$,

then B will only be ranked higher than A if $w_B > w_A + T$, where T is the switching threshold (defined below). An item k not in the previous list is assumed to have $r_k = \infty$.

In the original version of AccessRank, $T = \delta$, where $\delta \geq 0$ is an AccessRank parameter. Here the switching threshold is uniform for all items, making it relatively difficult for low weighted items to change positions, while making it relatively easy for higher weighted items to change. In a later version (AccessRank 2), $T = w_A\delta$, making the switching threshold relative to the weight of the items involved. In comparison to the original switching threshold, this makes it harder for highly ranked items to change positions, but easier for lower ranked items.

The switching threshold comparison is not transitive; for example, consider items A, B and C, where $r_A < r_B < r_C$. Let $\delta = 0.5$, $w_A = 1.1$, $w_B = 1.5$ and $w_C = 1.9$. Now, regardless of whether the original AccessRank or AccessRank 2 is used, $w_B < w_A + T$ and $w_C < w_B + T$, but $w_C > w_A + T$. The prediction list will differ depending on the comparison order, in turn determined by which sorting algorithm is used. Merge sort was used in the evaluated implementations of AccessRank. Variations that satisfied the transitivity property did not perform as well.

6.3 Converting Logs for Analysis

To evaluate AccessRank against other prediction algorithms, several log datasets were collected and converted to a standard form ready for analysis:

Window switching Logs were used from a longitudinal study on window switching behaviour run by Tak [172].

Web browsing Logs were used from a 6 week web browsing study by Tauscher and Greenberg [174].

UNIX logs Logs were used from command-line usage data from a study by Greenberg involving 168 *csh* users [84].

File browsing Logs were used from the longitudinal study on file browsing described in Chapter 5.

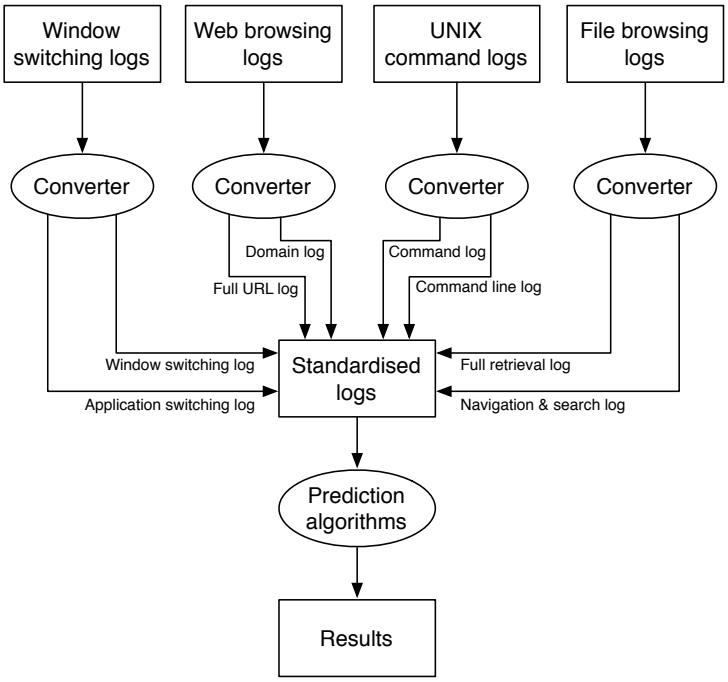


Figure 6.1: Process to compare ranking algorithms

Figure 6.1 shows a high level overview of how the algorithm comparison was done. The first step was to convert the logs into a standardised format, as each was in a format specific to its study. Each dataset was converted into two sets of logs based on particular semantics of the domain: window switching into logs of window switches and application switches, web browsing into logs of website visits (full URLs) and web domain visits, UNIX logs into command line logs and logs of just the command names used, and file browsing into full retrieval logs and logs of just retrievals performed through navigation and search. Details of this conversion process are provided below.

6.3.1 Standardised Log Format

In order to be able to easily compare different logs, they were converted to a standardised XML log format. Each log file is specific to a particular user. This log format consists of a list of log entries, with each log entry consisting of a set of key-value pairs.

Log entries are broken down into three primary events (visits, additions and removals) as well as an optional one (sessions):

Visit The main action in a log depends on the context. For example, in a window switching log this would be a window switch, while in a file browsing log it would be opening a file. While the specific action depends on the type of log, in the standardised log format they are all classed as *visits* to simplify processing.

Addition As the logs are used to evaluate prediction algorithms, a set of possible action targets must be maintained. The algorithms must assign probabilities to each possible target immediately before each *visit*. An *addition* adds a target to the set of targets so that it will have its own assigned probability. This would normally appear immediately after the first *visit* entry for a target, so that the target does not have an assigned probability when it is first observed.

Removal In some datasets, targets can be removed from the set of possible action targets. For example, for window switching logs, closing a window removes it from the set. Such an event results in a *removal* entry, resulting in the target no longer getting a specifically assigned probability. Note that in some cases, targets can be added and removed multiple times in a single log, for example in window switching logs when a document is opened and closed multiple times.

Session In datasets that explicitly record new sessions, a session event is added at the start of each session.

Additionally, the first log entry in each converted log is a special entry contains summary statistics, often specific to the type of log:

Log type The type of log, for example "Window switching".

Visits The number of *Visit* entries in the log.

Additions The number of *Addition* entries in the log.

Removals The number of *Removal* entries in the log.

SessionDataIncluded True if sessions are explicitly recorded in the log. Assumed to be false if omitted.

Domain specific entries Entries specific to the type of log, for example the number of items that were excluded during the conversion process for a particular domain specific reason.

All subsequent entries correspond to events, ordered by time, as described above. The structure of these entries is given below:

Type The type of log entry; "Visit", "Removal", "Addition" or "Session"

ID A unique identifier representing the object that was the target of the event, for example a window ID, file path, or website URL (depending on the log type). The ID property is excluded for *Session* entries.

Time The timestamp of the event.

Window Switching Logs

Tak [172] ran a three week longitudinal window switching study on Microsoft Windows with 25 participants. Logs from this study recorded window switches (changes in which window was topmost), window resize events, the cause of these events (i.e. mouse clicks, key presses or system events), and a list of all open windows at the time of each event. They also included both unique window identifiers, as well as the application owning the windows.

These logs were converted into two sets of logs: the first consisted of window switches, while the second consisted of *application* switches, ignoring switches between windows of the same application. The conversion process was otherwise the same, and is summarised below, interpreting a window switch as an application switch for the purposes of the application switching log:

- Window switches and new windows were generally recorded as a *Visit* event for the new window, with a few exceptions detailed below. Resize events were ignored.

- A window switch $W_1 \rightarrow W_2$ was ignored if W_2 appeared without user interaction, was a child window of another window, or was the login window. Additionally, the next window switch ($W_2 \rightarrow W_3$) was ignored if $W_1 = W_3$; if $W_1 \neq W_3$, it would still be processed, therefore appearing in the processed log as $W_1 \rightarrow W_3$.

- If a sequence of window switches $W_1 \rightarrow W_2, W_2 \rightarrow W_3, ..., W_{k-1} \rightarrow W_k$ occurred for $k \geq 3$ such that $T_{i+1} - T_i < 1$ for all $1 \leq i \leq k-2$, where T_i is the time, in seconds, at which the window switch $W_i \rightarrow W_{i+1}$ occurred, the sequence was reduced to $W_1 \rightarrow W_k$. In other words, window switches were ignored if another window switch occurred within a second, under the assumption that the incorrect window had been activated. If $W_1 = W_k$, no switches were recorded since a window switch $W_1 \rightarrow W_1$ would have no meaning.

- *Addition* events were added after observing a new window that had not been excluded for one of the reasons described above, and *Removal* events were added when a window was no longer included in the window list at the time of an event.

Web Browsing Logs

Tauscher and Greenberg [174] ran a six week study which recorded detailed usage data of a web browser. Although this dataset is relatively old (1997) and therefore possibly not descriptive of modern web browsing behaviour, it is still useful in a more general sense to evaluate prediction algorithms over a range of domains. The logs from this study included time, URL, information about actions performed and other information.

These were converted into two types of converted logs:

1. Logs of full URLs, where each URL was treated independently, regardless of domain or even if it was just a different anchor on the same page. If the same URL was visited twice in a row, it would still be included twice in the converted log.

2. Logs of domains only. Consecutive accesses of multiple pages within the same domain would be collated into one entry in the converted logs. Accessing a website over a different port counted as the same domain.

The log conversion process was similar for these two types, and is outlined as follows:

- If a log entry did not relate to opening a URL, it was ignored.
- For log entries that were not ignored, a *Visit* entry was added to the converted log, followed by an *Addition* entry if it was the first time the URL was observed. Such events included manually entering a URL, clicking on a link, or choosing an item from history.
- No *Removal* entries were added to the converted log as all URLs could potentially be revisited.

UNIX Logs

Greenberg [84] collected command-line usage data from 168 users of the UNIX *csh* command interpreter [109] in 1987. These users had varying levels of experience ranging from non-programmers to experienced programmers and computer scientists. The logs include each command line entered, the current working directory, any alias the command line invoked, if the history mechanism was used and any errors that occurred. They also indicate the start of each login session. The only timestamps included are the start times, and sometimes end times, for each session; there are no timestamps for individual commands entered.

As with the other datasets, two types of converted logs were generated. The first includes full command lines, while the second includes just the commands used. In both cases, consecutive repetitions of the same command or command line are included in the converted logs.

The log conversion process was similar to other log types such as the web browsing logs, whereby *Visit* entries were added for each command or command line executed and *Addition* entries were added following these if the command or command line had not previously been used by that user.

Since timestamps were not included for command uses, the login session's start time was used to approximate the timestamp for each command.

File Retrieval Logs

Chapter 5 described the results of a four week study of file retrieval. Logs from this study include a variety of information, include details of when and what files were opened.

These logs were converted into two sets; one containing all file retrievals, and another containing just those retrievals that were directly observed (i.e. through use of Finder navigation or search). The former set consists of a more complete, diverse set of retrievals, however those retrieved outside the directly observed methods were only observed after a slight time delay, meaning that some retrievals could be incorrectly ordered. Additionally, they may have been misclassified in some cases – see Chapter 5 for a full discussion. The latter set consists of a limited set that contains fewer revisitations per file, on average, since retrievals are not included after transitioning, for example, from navigation to a recent items interface. However, it is more suitable to assess predictive algorithms for use within a particular retrieval interface. The same assumptions and filters discussed in Chapter 5 were used; for example, entries on the first day of the study were omitted, and some indirectly observed retrievals were omitted when it was suspected that they were likely just activations of already open documents.

The log conversion process was again similar to other domains; *Visit* entries were added for each file retrieval and *Addition* entries were added following these for previously unobserved file paths. The direct-retrievals set counted every *OpenFile*, *QuickLook* and *SpotlightSelection* event as a retrieval, while the all-retrievals set also included *OpenFileExternal* events.

6.4 Measures Used to Compare Algorithms

Prediction algorithms can be compared using a range of measures, categorised as either *prediction measures*, which assess how well an algorithm is at predicting what will happen next, and *stability measures*, which assess how stable an algorithm's prediction lists are over time. Additionally, *data*

characterisation measures provide information about the datasets as a whole, rather than measuring the effects of an algorithm.

The sections below describe the particular measures used in the analysis of the algorithms.

6.4.1 Prediction Measures

The most obvious characteristic of a good prediction algorithm is that it ranks relevant items highly, however there are many ways to measure this. Those considered as part of this analysis are as follows.

Percentage Revisitations Predicted records the percentage of all revisitations that are at the top of the prediction list generated by an algorithm immediately prior to the event. Items that have not previously been accessed are not considered.

Average Rank records the average position of items in the prediction lists generated by an algorithm immediately prior to the visits. In cases where the item is not in the prediction list, for example if it has never previously been accessed, it is excluded from the calculation.

Percentage Revisitations Predicted With Hint is the same as *Percentage Revisitations Predicted*, but is calculated for varying hint sizes. This hint is the first characters of the identifier of the next access, after removing common prefixes such as 'http://' for web logs. This measure indicates how well the algorithm might work in a real world setting for interfaces that allow users to type in the start of the identifier for the item they are looking for (as in many web browsers).

Percentage Included In List records the percentage of revisitations that are included in prediction lists of size n generated by an algorithm, for varying values of n. This facilitates investigations of the distribution of list rankings for items and suitable list sizes for a dataset. When $n = 1$, this measure is identical to *Percentage Revisitations Predicted*.

6.4.2 Stability Measures

There are significant advantages to algorithms providing stability in the prediction lists they provide, to reduce the time users spend searching a list for a target item. If the item is at a consistent location, they will learn to quickly select it.

There are a number of measures to compare list similarity. Webber et al. classify these measures based on two main properties [181]:

Conjointness: Measures on conjoint rankings involve both lists consisting of the same items, but not necessarily in the same order. Non-conjoint rankings do not necessarily involve the same items and are also known as top-k rankings.

Weightedness: Unweighted measures give the same importance to an item regardless of rank in a list. Weighted measures typically give greater importance to higher ranked items.

For the purposes of user interfaces, top-weighted measures are of most relevance as the top of a prediction list is the most important; for example, the further down a list someone must look to find a relevant item, the more likely they are to give up, and many interfaces provide particularly quick access to the top item in a list (for example, many web browser URL suggestion interfaces). Additionally, since the set of items being ranked changes over time, it is important for measures to handle non-conjoint rankings. The first two stability measures below satisfy these properties, while the third, *Learnability*, is non-conjoint but unweighted.

While list similarity measures calculate the similarity of two lists, a full analysis of stability must measure how much the prediction lists change over time. Let P_i be the ranked prediction list for a particular algorithm that corresponds to the ith visit event in a log file with n total visits. Let $F(P_1, P_2)$ be a list similarity measure. A stability measure S can be determined as follows:

$$S = \frac{1}{n-1} \sum_{i=1}^{n-1} F(P_i, P_{i+1}) \tag{6.6}$$

This formula is used for each of the similarity measures outlined below. Note that for each measure, the list similarity score will generally be in the range $[0, 1]$, where 0 indicates no similarity and 1 indicates perfect similarity.

Average Overlap

Average overlap is a top-weighted non-conjoint measure of list similarity. Independently invented as *average accuracy* by Wu and Crestani [186] and as *intersection metric* by Fagin et al. [65], it is also known as weighted overlap [87] and average overlap [181].

Let S and T be two ranked lists, and let S_i be the i top ranked items of S. Let O_i be the number of items in both S_i and T_i and define agreement at depth i as $A_i = \frac{O_i}{i}$. The average overlap AO_k for S_k and T_k is calculated as follows:

$$AO_k = \frac{1}{k} \sum_{i=1}^{k} A_i \tag{6.7}$$

Webber et al. discuss several properties of average overlap that are disadvantageous [181]. Deeper evaluation can result in the average overlap moving in the opposite direction of what is intuitive, i.e. AO_k and A_k do not necessarily increase and decrease together as k increases. For example, $AO_{k+1} > AO_k$ if $A_{k+1} > AO_k$, regardless of whether the $(k+1)$th elements of S and T are in T_{k+1} or S_{k+1} respectively. Furthermore, they show that AO_k is non-convergent as $k \to \infty$, as its infinite tail always dominates the finite prefix that has been examined.

In the implementation used in this study, $k = \min(|S|, |T|, 20)$, as it would be a rare use case for more than 20 items to be evaluated in a predictive user interface before either filtering results or abandoning it for a more reliable method (such as file navigation or entering a more complete web URL).

Rank-Biased Overlap

Webber et al. created rank-biased overlap (RBO) [181], which solves the issues of average overlap described above. It does so by assigning decreasing weights at each depth, based on a parameter p where $0 < p < 1$. It is calculated as follows for lists S and T:

$$RBO(S,T,p) = (1-p)\sum_{d=1}^{\infty} p^{d-1} A_d \tag{6.8}$$

They also describe several formulae for estimating RBO values based on list prefixes. In particular, they include a calculation for an extrapolated point estimate of list similarity based on S_k and T_k:

$$RBO_{EXT}(S,T,p,k) = p^k A_k + \frac{1-p}{p}\sum_{d=1}^{k} p^d A_d \tag{6.9}$$

The implementation used for this study uses $k = \min(|S|, |T|, 50)$ and $p = 0.9$, a configuration which Webber et al. suggest gives an almost exact estimate of the true RBO score. Later references to Rank-Biased Overlap or RBO refer exclusively to the RBO_{EXT} score.

Learnability

Learnability is an estimate of stability developed by Cockburn et al. to measure how easy it is to remember item locations in a list over time [46]. They describe it as follows: "[Learnability] can be estimated for different interfaces by calculating one minus the average distance that items move as a proportion of half of the total menu length – e.g., random items will on average move half of the menu length l per selection, hence $L = 1 - \frac{0.5l}{0.5l} = 0$."

In the study implementation, prediction lists were kept to a maximum size of 10. While suitable list sizes depend on the domain, a fixed maximum ensured that comparisons between algorithms and datasets were valid, and is a suitable compromise for the lack of weightedness in the measure; people are more likely to use an alternative approach such as providing a larger hint if the result is not near the top of the list. To satisfy the non-conjoint property, items that were in only one list were treated as having a distance moved of half the total menu length. The average distance was taken across all items that appeared in at least one list.

6.4.3 Data Characterisation Measures

While the above measures are calculated for each algorithm, the measures below are independent of the algorithms and help characterise the data.

Revisitation Rate is the percentage of accesses that are of items previously accessed. Alternatively, subtracting this rate from 100% gives the percentage of accesses that are to previously unvisited items.

Time Between Visits is the average time between visit events.

Number of Sessions is calculated in one of two ways. If the original logs include session information, the number of sessions is just tallied from this data. Otherwise, as an estimate, a new session starts when the time between two visits exceeds two standard deviations of the mean; the large skew in the data (with clumping within a session) ensures that this is normally an appropriate time requirement. This is calculated using the results of the *Time Between Visits* measure, and used by the *Adaptive* algorithm.

6.5 Analysis of Algorithm Performance

Once the log datasets were converted to a standard format, algorithm performance was analysed with a specialised log processor. This takes as input a series of datasets, and it dynamically loads prediction algorithms and evaluates them against various measures. At a high level, log processing is done as follows:

1. Dynamically load algorithms and measures based on a configuration file

2. For each subject in each dataset:

   (a) For each data characterisation measure:

      i. Iterate through the events in the subject's log file, and update the data characterisation measure after each one.

(b) For each event in the subject's log file:

i. For each prediction algorithm:

A. Make predictions

B. Update each accuracy and stability measure based on predictions

Algorithms from a variety of domains, described earlier in this chapter, were used in the analysis. The factors under study were as follows:

Algorithm $\in$ {*MRU, MFU, Adaptive, Places Frecency, Split Recency & Frequency, Combined Recency and Frequency, six AccessRank variations* ($\lambda = 0.8$ *and* 1.65, $\delta = 0$, 0.2 *and* 0.5) *and 15 AccessRank 2 variations* ($\lambda = 0.8$, 1.65 *and* 2.5, $\delta = 0.5$, 1, 2, 3 *and* 5)}

Log dataset $\in$ {*window switching, application switching, web URLs, web domains, Unix command lines, Unix commands, file retrievals (directly observed), file retrievals (all)*}

6.5.1 Results

The results of the log processing simulation are provided below. The results first give a brief overview of dataset revisitation properties. Next, accuracy and stability results are described independently, after which the combination of the two measures is analysed in detail. The remainder of the results provide more in-depth analysis on differences in predictive properties of the datasets under investigation and properties of AccessRank scores that allow for future expansion of the algorithm.

Revisitations

The level of revisitation differed substantially by dataset. For file retrieval data, 48% of directly observed visits revisited a previously observed file, while 55% of all visits (directly or indirectly observed) were revisitations. For web data, 26% of visits were revisiting an exact URL, while 43% were revisiting a domain. For window switching data, 51% of visits were revisiting a window,

while 78% were revisiting an application; for this data, closing and reopening a window or application did not count as a revisitation. For UNIX data, 72% of command lines were exactly the same as a previous line, while 93% invoked a previously used command.

Accuracy

Average Rank and *Percentage Revisitations Predicted* were the primary measures of accuracy. The former is of particular importance for ranked interfaces that are a primary access mechanism, as it is important that items are accessible and minimising the average search distance is essential. The latter is of greater importance for supplementary interfaces that are used mainly for commonly accessed items, especially when accessing the top result is substantially easier than other results (for example, in an auto-completion interface).

Tables 6.1 and 6.2 show the results for *Average Rank* and *Percentage Revisitations Predicted*, respectively. It is clear from these that the algorithms vary considerably by dataset, and that no algorithm is superior in all datasets. However, the Markov algorithm and AccessRank with parameters $\lambda = 1.65$ and $\delta = 0$ generally performed best in terms of accuracy alone (note that AccessRank and AccessRank 2 are identical when $\delta = 0$).

Accuracy was also affected by the size of prediction lists and by providing a 'hint' of varying size – that is, when filtering the predictions based on a known prefix of the next item's title. An analysis of this was limited to one domain per dataset: window switching, Unix command lines, directly observed file retrievals and web URLs. Figure 6.2 shows the percentage of all revisitations for which the revisited item occurs in a list of size n. Results show that even without any hint as to what the user wants to do next, there is a reasonable chance of the desired item being placed highly in a prediction list. There are differences between domains in terms of how well some algorithms perform; for example, Most Recently Used is worst for Unix commands and for the top prediction for window switching, but is amongst the best for file retrievals and for longer prediction lists for window switching.

A similar pattern emerges for hint size, shown in Figure 6.3, with the

Algorithm	Avg.	File	File*	Win	App	CmL	Cmd	URL	Dom
MRU	19.3	24.0	42.8	2.1	2.2	32.2	9.0	32.4	10.1
MFU	26.7	63.1	65.6	2.9	2.6	30.1	8.5	32.7	8.4
Adaptive	25.4	37.6	60.3	2.9	2.6	46.4	9.1	35.0	9.2
Places Frecency	25.4	59.4	62.7	2.9	2.6	27.2	8.1	31.7	8.2
Markov	**15.4**	**18.0**	**36.7**	**1.8**	1.7	26.0	6.3	**24.3**	8.5
CRF (0.1)	18.3	24.5	42.7	2.0	2.1	27.4	7.1	31.7	8.9
AccessRank (0.8, 0)	17.7	23.8	42.0	1.9	1.7	26.5	6.5	30.7	8.4
AccessRank (1.65, 0)	16.9	22.5	40.6	1.8	**1.7**	25.5	6.1	28.8	**7.9**
AccessRank (0.8, 0.2)	18.2	24.6	42.8	1.9	1.8	27.4	7.1	31.7	8.6
AccessRank (1.65, 0.2)	18.9	25.7	43.7	1.9	1.8	28.4	8.4	32.5	8.9
AccessRank (0.8, 0.5)	19.6	26.6	45.4	1.9	1.8	29.8	8.6	33.5	9.1
AccessRank (1.65, 0.5)	20.9	30.1	47.0	2.1	2.0	30.1	9.7	36.0	9.9
AccessRank 2 (0.8, 0.5)	17.9	24.6	42.3	1.9	1.8	26.5	6.5	31.3	8.5
AccessRank 2 (0.8, 2)	18.6	26.1	43.1	2.1	2.0	26.7	6.7	32.7	9.1
AccessRank 2 (0.8, 5)	19.2	27.7	44.1	2.2	2.1	27.0	7.0	34.2	9.7
AccessRank 2 (1.65, 0.5)	17.3	24.1	41.2	1.9	1.8	25.4	6.1	30.0	8.4
AccessRank 2 (1.65, 2)	18.6	27.0	42.7	2.1	1.9	25.8	6.2	33.2	9.5
AccessRank 2 (1.65, 5)	19.9	30.6	44.7	2.3	2.1	26.4	6.5	36.1	10.3
AccessRank 2 (2.5, 0.5)	16.9	23.5	40.1	1.9	1.8	**24.7**	**5.9**	29.0	8.3
AccessRank 2 (2.5, 2)	18.6	27.6	42.2	2.2	2.0	25.1	6.0	33.4	10.1
AccessRank 2 (2.5, 5)	20.2	31.8	44.6	2.4	2.1	25.7	6.2	37.7	11.0

Table 6.1: Average rank of items in prediction lists. Dark grey denotes the algorithm with the best rank, while light grey denotes the second best. Dataset abbreviations are *File=Directly observed file retrievals*, *File*=All file retrievals*, *Win=Windows*, *App=Applications*, *CmL=Command lines*, *Cmd=Commands*, *URL=Web URLs*, *Dom=Web domains*.

Algorithm	Avg.	File	File*	Win	App	CmL	Cmd	URL	Dom
MRU	15.6	26.0	11.9	30.1	18.0	8.5	14.0	15.9	-
MFU	19.3	7.6	8.3	30.3	32.9	17.6	22.1	12.1	23.3
Adaptive	19.4	8.5	8.6	30.4	33.0	17.7	22.0	11.8	23.5
Places Frecency	19.5	7.8	8.6	30.3	32.9	18.3	22.3	12.2	23.5
Markov	36.8	**38.1**	**23.8**	57.4	63.3	29.9	35.4	**27.4**	19.1
CRF (0.1)	23.9	16.3	11.2	43.4	37.3	23.2	25.2	13.8	21.0
AccessRank (0.8, 0)	33.9	22.6	15.8	55.1	62.5	33.5	35.7	17.4	28.5
AccessRank (1.65, 0)	**37.3**	27.8	19.0	**60.3**	**64.1**	**36.0**	**37.6**	21.5	**32.1**
AccessRank (0.8, 0.2)	33.1	21.1	15.0	54.8	61.4	32.8	35.1	15.4	29.1
AccessRank (1.65, 0.2)	33.8	25.0	16.0	55.8	59.7	32.9	34.5	17.2	29.0
AccessRank (0.8, 0.5)	31.7	19.8	14.0	52.9	59.8	31.4	33.9	13.5	28.3
AccessRank (1.65, 0.5)	29.8	20.8	13.5	50.5	54.6	29.8	30.0	13.1	25.8
AccessRank 2 (0.8, 0.5)	31.7	19.5	14.6	52.1	58.7	32.2	34.1	14.3	27.9
AccessRank 2 (0.8, 2)	27.2	14.9	12.1	46.3	51.8	28.5	29.8	11.2	23.1
AccessRank 2 (0.8, 5)	24.5	12.2	10.5	41.7	46.2	25.3	27.0	10.5	22.5
AccessRank 2 (1.65, 0.5)	35.1	26.1	18.0	56.0	59.4	35.5	37.2	18.4	30.3
AccessRank 2 (1.65, 2)	31.0	21.1	15.3	47.8	52.6	32.9	35.3	16.0	27.0
AccessRank 2 (1.65, 5)	26.9	15.6	12.5	42.1	47.7	30.1	31.9	12.5	22.5
AccessRank 2 (2.5, 0.5)	35.1	28.7	19.7	54.7	57.7	35.2	37.2	19.4	28.6
AccessRank 2 (2.5, 2)	32.1	26.7	18.2	47.1	52.0	33.3	36.4	18.8	24.4
AccessRank 2 (2.5, 5)	29.6	21.0	15.6	43.3	48.5	32.0	34.8	15.6	25.7

Table 6.2: Percentage of the time revisitations were the top prediction of an algorithm. Dark grey denotes the algorithm with the best accuracy, while light grey denotes the second best. Dataset abbreviations as in Table 6.1.

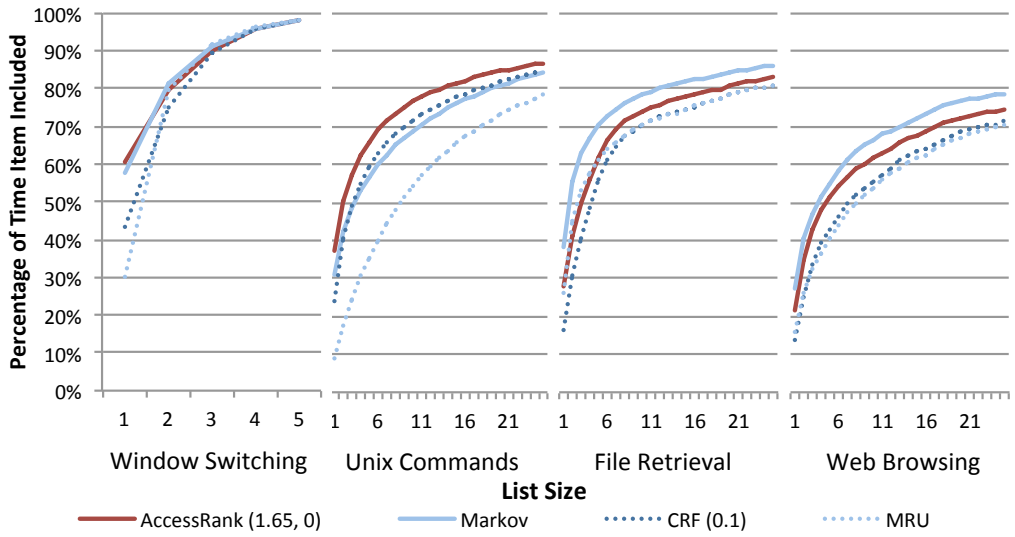


Figure 6.2: Percentage of revisitations that are included in a prediction list of a given size, across three datasets.

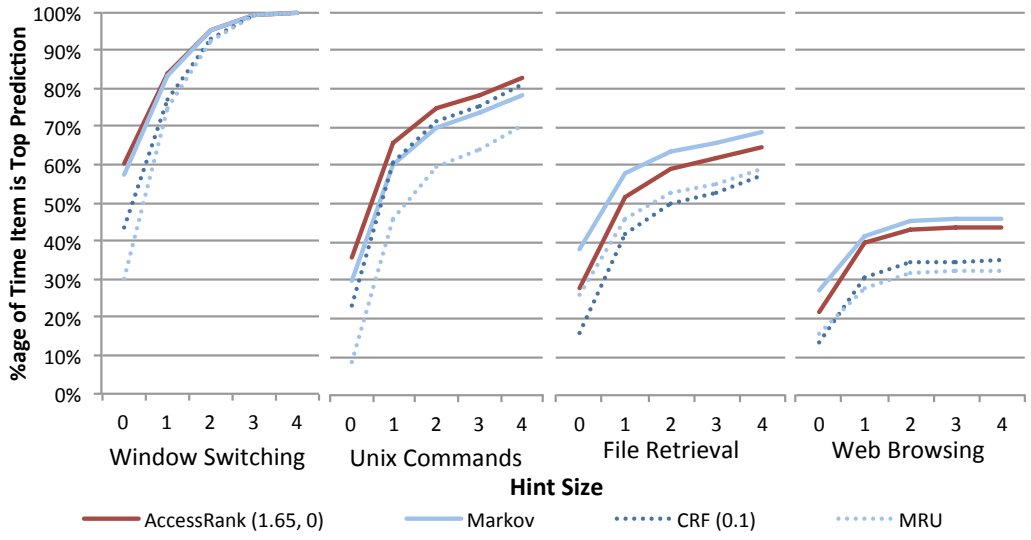


Figure 6.3: Percentage of revisitations that are the top match when filtered by a prefix of a given size, across three datasets.

AccessRank and Markov algorithms generally performing best. For all algorithms, almost all windows can be immediately acquired with a three character hint. For file retrievals, the hint was used to match the filename, rather than full path, as many paths would have the same common prefix. However,

this meant that some files would never be the top prediction regardless of hint size (for example, when multiple files have the same filename). For web browsing, even with a four character hint, less than half of revisitations can be predicted by any algorithm. This is likely because while a domain can be quickly determined using a hint, it takes a substantially larger hint size to deduce any information about what *page* on that domain is of interest. Other approaches, such as using the first character of each URL component as a hint, may be more effective, though less intuitive.

The Effect of History Size on Accuracy

Algorithms' individual properties affect how prediction accuracy changes with the size of the access history. For example, some algorithms, such as *Most Recently Used*, rely almost exclusively on recent accesses, while others continue to improve as they gain new information.

Figure 6.4 shows the effect of access history size on prediction accuracy for file navigation retrievals. As expected, *MRU* quickly reaches and maintains its peak level of accuracy. *MFU*, however, initially performs well but quickly deteriorates in performance. This is likely due to changes in the most commonly accessed files over time, combined with *MFU*'s lack of time-based weighting. Initially, the most frequently visited items are *also* recently visited, whereas this is not true as the history size increases. *CRF*, which does include a time weighting, does not deteriorate as much.

The *Markov* algorithm continues to improve in accuracy as the amount of information available to it increases. This is consistent across datasets. AccessRank variations, which incorporate both the *Markov* and *CRF* algorithms, show a similar pattern, though show a slightly smaller improvement than the *Markov* algorithm due to the latter's influence. Algorithm performance, as observed across different datasets, varies little when history sizes increase beyond about 2000 accesses.

These results, in conjunction with the earlier accuracy results, indicate that recency is a much stronger predictor than frequency in the domain of file navigation retrievals. Algorithms that weigh recency more highly are likely to outperform others, and hybrid algorithms such as *CRF* and *AccessRank* should be configured accordingly.

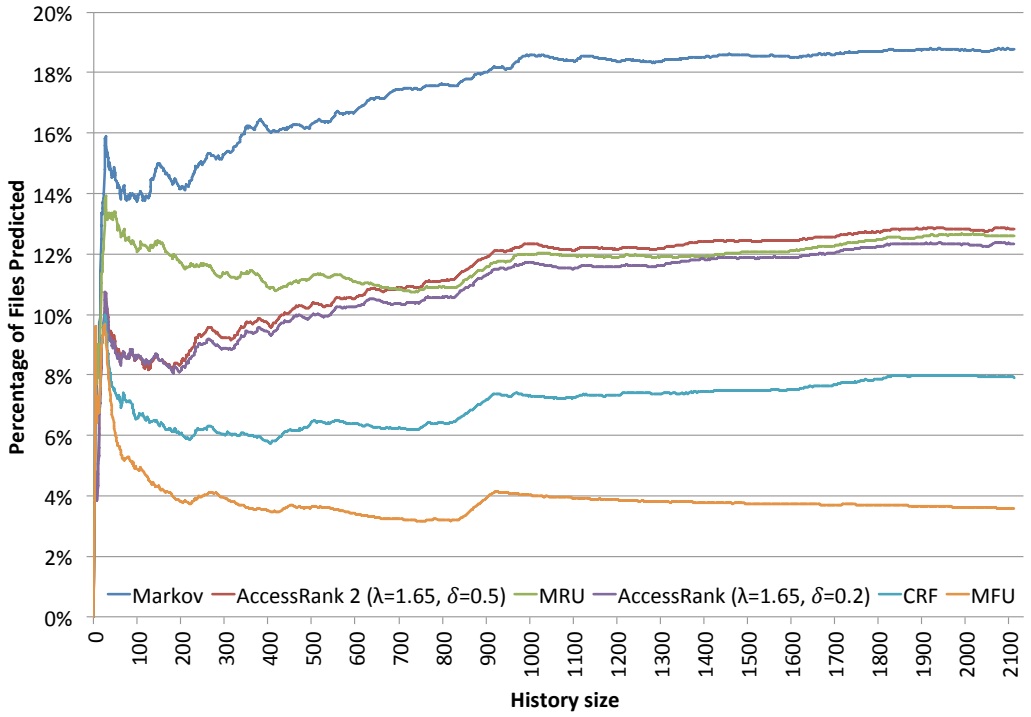


Figure 6.4: Variation in algorithm accuracy for file navigation retrievals based on the size of the recorded retrieval history. Accuracy is measured by the percentage of retrievals that match the top prediction of the algorithm.

Stability

The three stability measures (Average Overlap, Rank-Biased Overlap and Learnability) all gave similar results. Average Overlap and Rank-Biased Overlap ranked the algorithms in an identical order, while the Learnability algorithm ranks were on average less than one different. All three measures ranked *Most Frequently Used* as most stable, followed by the *Places Frecency* algorithm, and ranked the *Markov* algorithm as the most unstable. Stability for AccessRank and AccessRank 2 was low when δ was set to 0, but increased significantly for larger deltas. Similarly, low values of λ (indicating greater reliance on *CRF* compared to *Markov* weights) corresponded to high stability for AccessRank 2, although the trend is less clear for the original AccessRank.

As all three measures produced similar results, the rest of the analysis only considers *Rank-Biased Overlap* (RBO). This measure was selected as it

Algorithm	Avg.	File	File*	Win	App	CmL	Cmd	URL	Dom
MRU	0.84	0.81	0.79	0.85	0.87	0.94	0.96	0.77	0.77
MFU	**0.97**	**0.98**	**0.98**	0.93	**0.97**	**0.996**	**0.997**	0.97	0.94
Adaptive	0.97	0.96	0.97	0.93	**0.97**	0.993	0.996	0.96	0.94
Places Frecency	0.97	0.98	0.98	0.93	**0.97**	0.995	0.996	0.96	0.94
Markov	0.67	0.74	0.62	0.82	0.80	0.49	0.62	0.70	0.61
CRF (0.1)	0.90	0.88	0.86	0.90	0.95	0.92	0.95	0.84	0.89
AccessRank (0.8, 0)	0.86	0.86	0.83	0.88	0.89	0.85	0.87	0.83	0.85
AccessRank (1.65, 0)	0.79	0.82	0.77	0.86	0.84	0.73	0.78	0.80	0.74
AccessRank (0.8, 0.2)	0.91	0.90	0.90	0.90	0.91	0.93	0.93	0.89	0.92
AccessRank (1.65, 0.2)	0.92	0.90	0.91	0.90	0.92	0.93	0.95	0.90	0.91
AccessRank (0.8, 0.5)	0.94	0.93	0.94	0.91	0.92	0.95	0.96	0.93	0.95
AccessRank (1.65, 0.5)	0.95	0.94	0.95	0.93	0.94	0.96	0.98	0.95	0.96
AccessRank 2 (0.8, 0.5)	0.91	0.91	0.90	0.91	0.92	0.91	0.92	0.91	0.93
AccessRank 2 (0.8, 2)	0.95	0.95	0.94	0.93	0.94	0.96	0.96	0.94	0.96
AccessRank 2 (0.8, 5)	0.96	0.96	0.96	**0.94**	0.96	0.97	0.98	0.96	**0.97**
AccessRank 2 (1.65, 0.5)	0.86	0.89	0.86	0.89	0.87	0.82	0.82	0.90	0.86
AccessRank 2 (1.65, 2)	0.92	0.94	0.93	0.92	0.91	0.91	0.89	0.95	0.94
AccessRank 2 (1.65, 5)	0.95	0.96	0.96	**0.94**	0.93	0.95	0.94	**0.97**	0.97
AccessRank 2 (2.5, 0.5)	0.83	0.88	0.82	0.89	0.86	0.75	0.77	0.89	0.80
AccessRank 2 (2.5, 2)	0.89	0.92	0.89	0.91	0.89	0.84	0.83	0.94	0.89
AccessRank 2 (2.5, 5)	0.93	0.95	0.93	0.93	0.91	0.90	0.88	0.96	0.95

Table 6.3: RBO values of prediction lists. Dark grey denotes the algorithm with the highest stability, while light grey denotes the second most stable. Dataset abbreviations as in Table 6.1.

is top-weighted and does not feature the previously described deficiencies of Average Overlap. It is worth noting, however, that the original AccessRank performs relatively better with the *Learnability* measure, since its use of the δ parameter is biased towards stability at low ranked list positions, which are weighted less by the other two stability measures. Table 6.3 summarises RBO results across different datasets.

Comparing Accuracy and Stability

While it is useful to investigate accuracy and stability independently, prediction algorithms used in user interfaces are most useful when they have high scores in both. For the purposes of this section, *performance* refers to a combination of high accuracy and high stability.

Figure 6.5 shows the relationship between *Rank Biased Overlap* and *Average Rank* averaged over all datasets, while Figure 6.6 shows the relationship between *Rank Biased Overlap* and *Percentage Revisitations Predicted*. AccessRank variants perform well in both cases, though particularly so for the latter. This is advantageous as it indicates that a strength of AccessRank

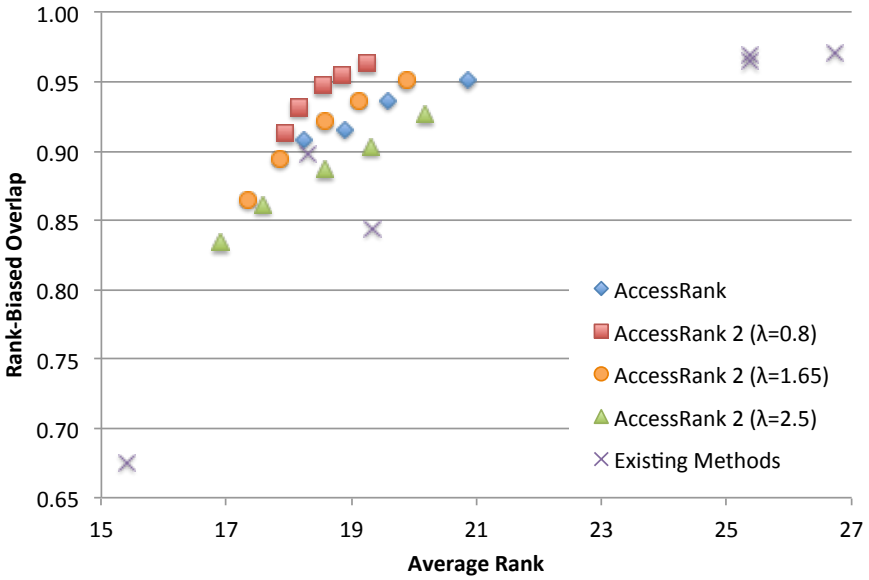


Figure 6.5: RBO versus Average Rank, averaged over all datasets. AccessRank 2 used $\delta \in \{0.5, 1, 2, 3, 5\}$ (shown from low to high RBO values). AccessRank parameters are $\lambda \in \{0.8, 1.65\}$, $\delta \in \{0.2, 0.5\}$.

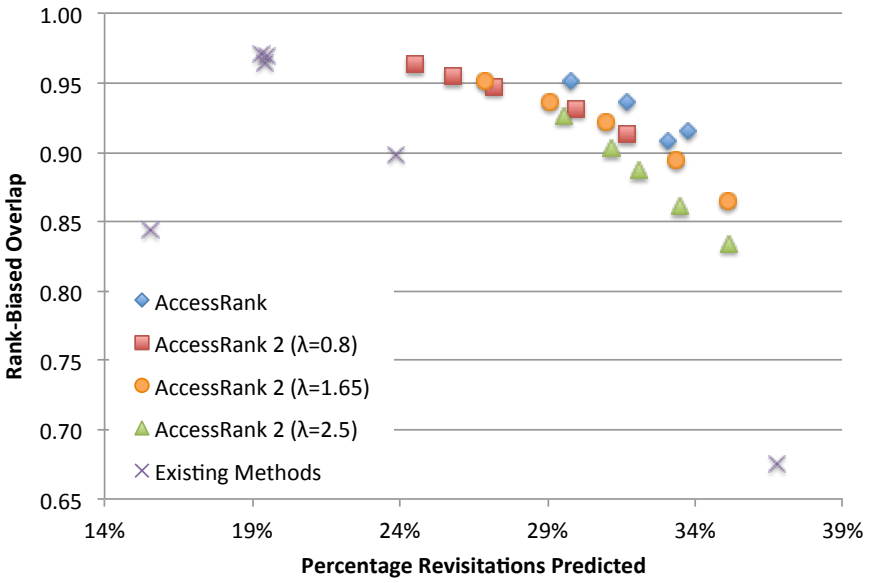


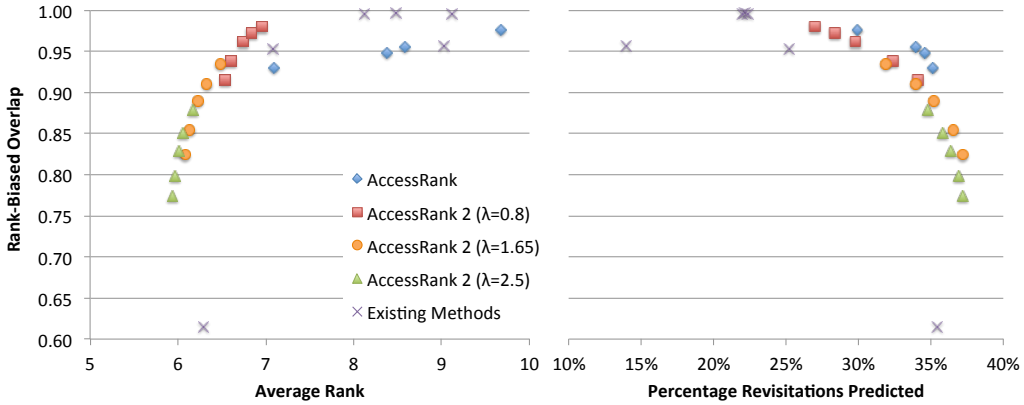
Figure 6.6: RBO versus Percentage Revisitations Predicted, averaged over all datasets. AccessRank parameters are as in Figure 6.5.

is correctly determining the most likely item to be accessed next, which is particularly useful in many user interfaces as users need not spend time or keystrokes selecting alternative items. When comparing RBO and Average Rank, the *Combined Recency and Frequency* algorithm also performs well.

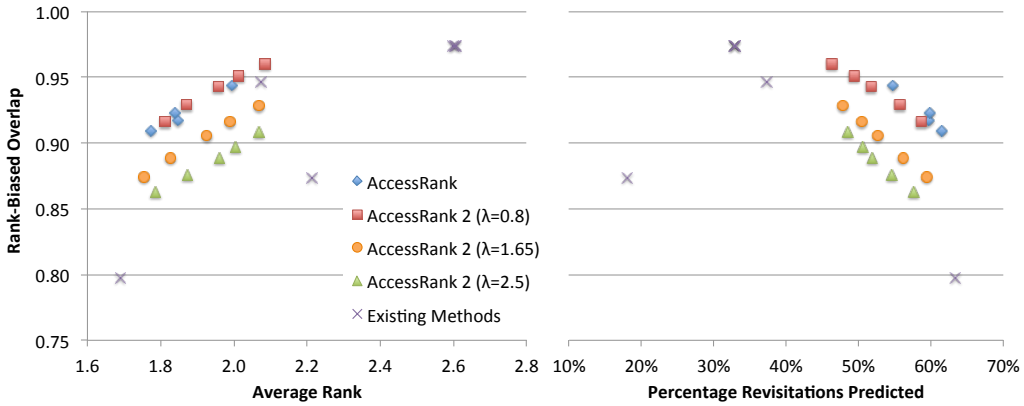
Notably, the relative performance of AccessRank and AccessRank 2 vary based on the accuracy measure. AccessRank outperforms AccessRank 2 when using *Percentage Revisitations Predicted*, likely because its use of the δ parameter makes rank changes relatively easier for highly ranked items compared to lowly ranked items. However, when *Average Rank* is considered, some AccessRank 2 configurations outperform AccessRank (in particular, with $\lambda = 0.8$). Decisions on which AccessRank variant to use should therefore depend on the type of interface – i.e. whether the top prediction or average rank are of most importance.

The graphs also illustrate the flexibility of AccessRank, in that it can be tailored to optimise either accuracy or stability as required, while still outperforming other algorithms. In particular, AccessRank 2 allows for more predictability about how its parameters affect the tradeoff between accuracy and stability. In effect, the λ parameter determines the shape of the accuracy-stability tradeoff curve, while δ determines the position on the curve. Higher values of λ (indicating greater influence of the Markov component) generally allow for better accuracy, however accuracy deteriorates more quickly as δ increases relative to increases in δ at lower values of λ. Low values of λ are thus more suitable when stability is important.

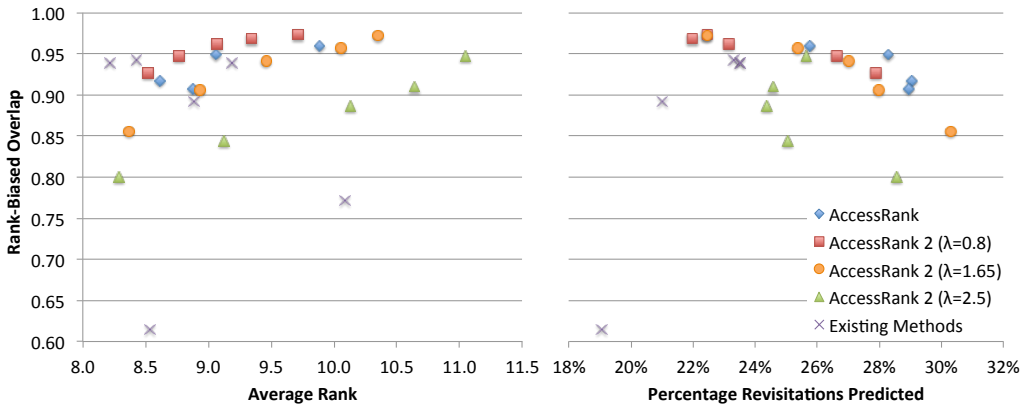
While the above discussion relates to average performance over all datasets, there was substantial variation between them, indicating that parameter calibration is best performed in a domain-specific context. Figure 6.7 shows comparisons for command usage, application switching and web domains, each showing noticeably different patterns. For command usage, there was little different between AccessRank 2 λ-curves, although lower λ values still corresponded to greater stability. AccessRank 2 significantly outperformed AccessRank using the *Average Rank* measure (though not using *Percentage Revisitations Predicted*). For application switching, higher AccessRank 2 λ values offered little, if any, accuracy advantage, and AccessRank performed as well or better than the top AccessRank 2 configurations regardless



(a) Commands

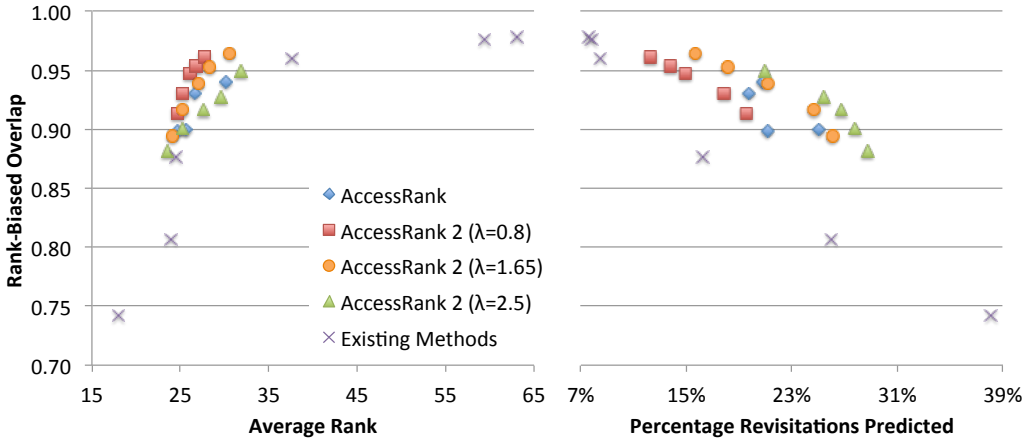


(b) Application switching

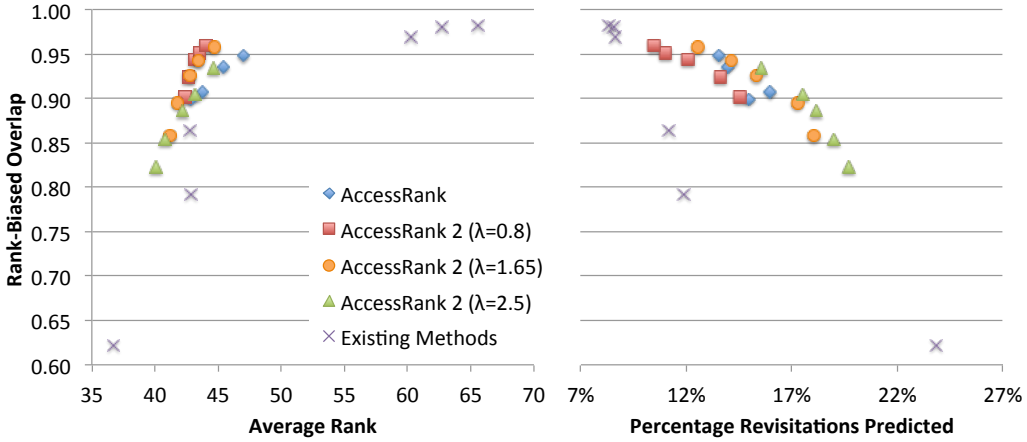


(c) Web domains

Figure 6.7: Accuracy vs stability for various domains. AccessRank parameters are as in Figure 6.5.



(a) Directly observed file retrievals



(b) All file retrievals

Figure 6.8: Accuracy vs stability for file retrievals. AccessRank parameters are as in Figure 6.5.

of accuracy measure. For web domains, AccessRank variations offered little performance advantage using the *Average Rank* measure, with some other algorithms outperforming them. However, they offered greater benefits for the top prediction. Notably, the adapted *Places Frecency* algorithm had the lowest average rank for web domains while still maintaining high stability, highlighting its suitability for use in web browsers (its actual performance is likely higher still since the adapted version was stripped of domain-specific considerations such as the effect of different access measures). However, it was outperformed in predicting the top result, an important feature in URL

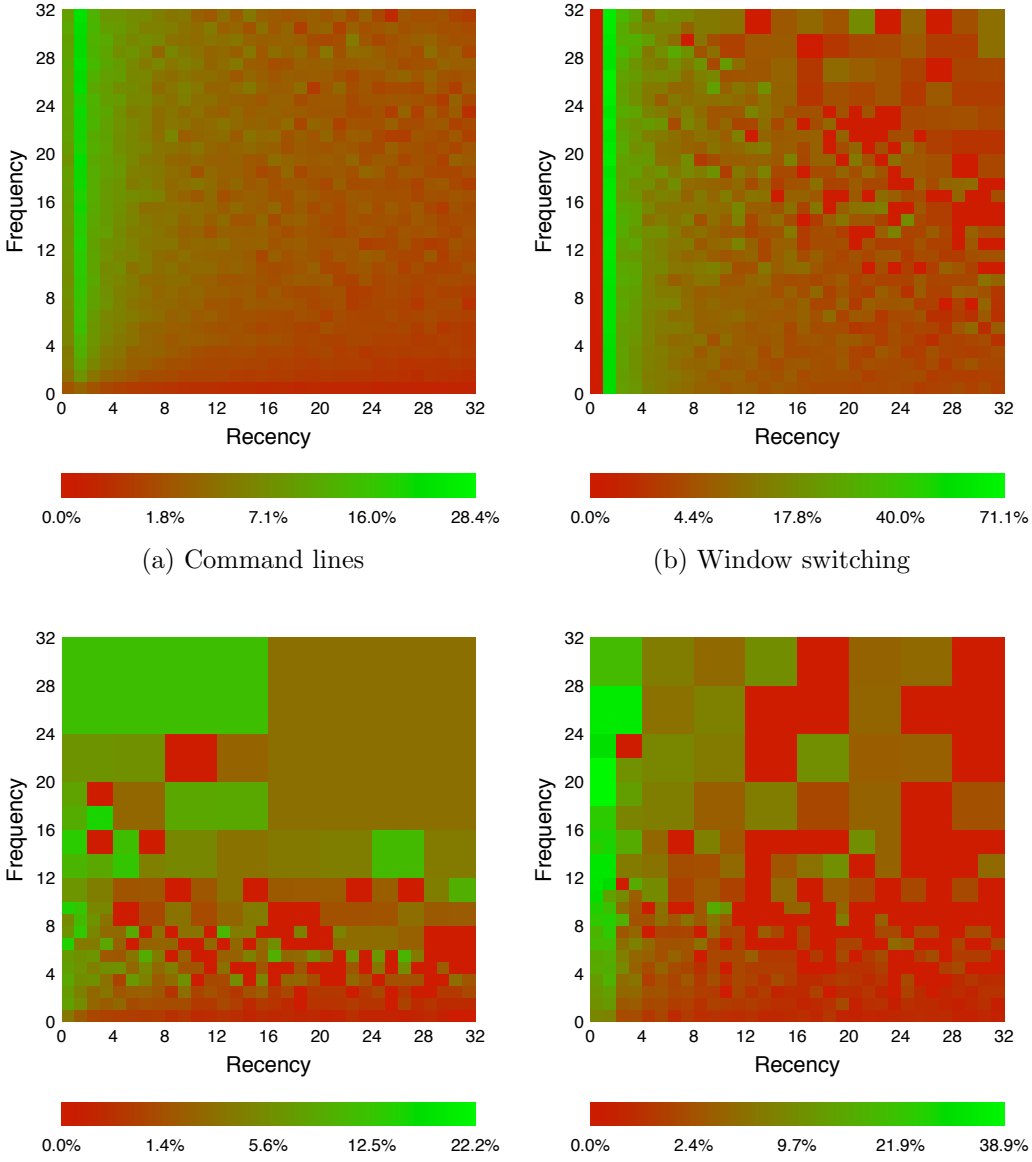
suggestion interfaces, and it did not perform as well for predicting full URLs as opposed to just web domains.

Figure 6.8 shows accuracy vs stability comparisons for both directly observed file retrievals and all file retrievals. AccessRank variants did not perform as well as other algorithms relative to other domains, but still showed modest improvements over existing algorithms when considering the accuracy-stability tradeoff. Both the *Markov* and *MRU* algorithms achieved high accuracy for directly observed files (albeit with low stability), indicating the importance of recency as a predictor for file retrievals.

Variation in the Predictive Power of Recency and Frequency

Differing performance of algorithms such as *MRU* and *MFU* indicated the presence of different predictive properties in each domain. This was confirmed by visualising the effect of recency and frequency on the probability that an item will be accessed next, shown in Figure 6.9. Each cell of the visualisations represents the observed probability that an item will be accessed next given both the number of times it has previously been accessed and the number of events since its last access.

There are noticeable differences between domains. In general, there are clear influences of both recency and frequency, although the influence of recency is typically greater. For window switching (Figure 6.9b), the influence of frequency is less clear. Importantly, there is a quick drop-off in accuracy for less recent items for directly observed file retrievals (Figure 6.9d), especially for those with low frequencies. The implication of this is that recency is an important predictor for file retrievals, as indicated in the previous section, but that its benefit as a predictor quickly diminishes as further retrievals occur. Also of note is that for some domains, such as command lines (Figure 6.9a), the second most recent item is a better predictor of the next item than the most recent item. The same is the case for window switching, but in that case it is because it would be impossible to switch to the same window twice in a row without switching to an intermediate window.



(a) Command lines

(b) Window switching

(c) Web URLs

(d) Directly observed file retrievals

Figure 6.9: The effect of recency and frequency on the probability that an item will be accessed next, across several domains. Recency is expressed as the number of accesses since the item's previous access. All cells contains at least 20 samples; those with fewer are automatically merged into larger ones. Colour spectrum uses a quadratic scale.

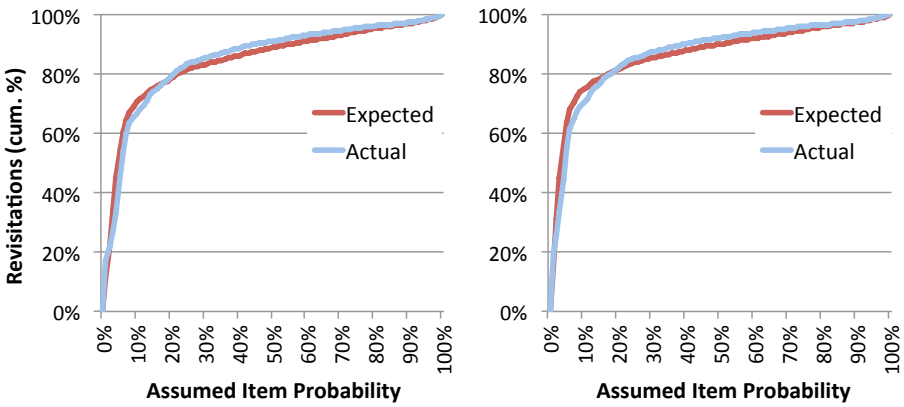
AccessRank Scores as an Estimate of Probability

AccessRank produces a score for each previously visited item that is indicative of the likelihood that that item will be the next accessed item. However, in some situations it is useful to map this score onto an actual probability. For example, potential AccessRank extensions might incorporate other factors not considered by the core algorithm. Supposing one of these factors doubles the likelihood that a particular item will be accessed, how should its score be modified? The score could be doubled, however this would not be appropriate if, for example, a doubling of an AccessRank score corresponded in general with only a 50% increase in probability.

To aid in future extensions, the following analysis tests the assumption that *an item's AccessRank score, in relation to scores of other items at a particular point in time, is proportional to the probability it will be accessed next.* This was tested with a simulation of each dataset. Before each revisitation, the AccessRank score of each item in the dataset's set of possible action targets was converted to an *assumed probability* by dividing its score by the sum of all scores.

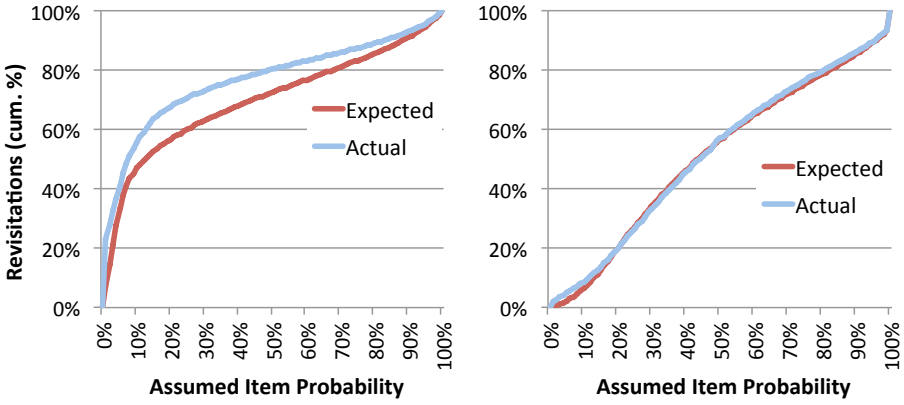
Next, the simulation collected data on the likelihood, according to the assumed probabilities, of the items that were actually accessed. These are shown in the graphs in Figure 6.10 as a cumulative percentage of revisitations across item probability. For example, in Figure 6.10a, the 'Actual' line shows that about two third of revisitations of files within the Finder were of items with assumed probabilities of less than 10%, whereas only 9% of window revisitations were in the same category (Figure 6.10d).

If the tested assumption is correct, the distribution of assumed probabilities of retrieved items should match the expected distribution of such probabilities for a particular dataset. For example, if a dataset averages 50 items each with a 1% chance of being accessed, and 1 item with a 50% chance of being accessed, it would be expected to observe as many accesses of items with a 1% probability than of items with a 50% probability, since $50 \times 1\% = 1 \times 50\%$. To determine the expected distribution of probabilities, the simulation recorded the assumed probabilities of *all* items at each point in time, not just those which were actually accessed.



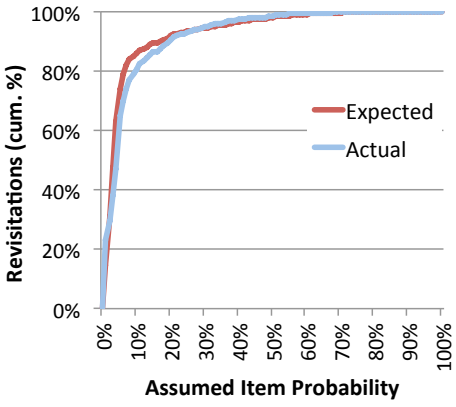
(a) File retrievals within the Finder

(b) All file retrievals



(c) Command lines

(d) Window switching



(e) Web URLs

Figure 6.10: Expected versus actual retrievals when assuming that AccessRank scores determine the probability of revisitation.

Figure 6.10 compares these expected and actual probability distributions. Notably, they are close to each other for file retrievals (either when limited to just those retrievals performed in a file browser, or when including all file retrievals), window switching, and web URLs. The distributions differ, however, for command lines, where low probability items are revisited more often than predicted by items' assumed probabilities. These results show that for most domains, AccessRank scores, as a proportion of total scores across all items, provide a reasonable approximation of the probability an item will be next visited. This result allows for simple extensions to the algorithm when additional factors have known effects on item probabilities. However, exceptions exist, and the assumption should be verified with data from the domain or domains in question, if not shown here.

6.6 Discussion

AccessRank is a flexible algorithm that outperforms existing algorithms in a variety of contexts when considering both accuracy and stability. There was significant variation between domains and, ideally, it should be calibrated within a domain before using it. However, when this cannot be done, the following general recommendations apply:

- When designing for user interfaces where both high accuracy and high stability are necessary, AccessRank (with $\lambda = 1.65$ and $\delta = 0.2$ or similar) or AccessRank 2 (with $\lambda = 0.8$ and δ between 1 and 5, depending on the exact accuracy/stability tradeoff required) are best. The AccessRank 2 configurations are best when minimising the average list position is the most important measure of accuracy, while the AccessRank configurations are best when maximising the accuracy of the top prediction is more important.

- When stability is of no importance, and accuracy of the top prediction is most important, setting λ to 1.65 or 2.5 and δ to 0 usually results in the best accuracy.

The sections below discuss potential improvements to AccessRank, as well as some possible applications.

6.6.1 Improving AccessRank

While AccessRank is a powerful algorithm, there are still improvements that can be made to it. Comparisons of results with and without its time weighting suggest that while beneficial, these benefits are relatively minor. The weighting is currently based exclusively on the absolute time, but might be more powerful if, for example, events were considered relative to when the computer is first used for the day. Additionally, location awareness could be incorporated to improve predictions; for example, if a user is at work, they are likely to access different items than if they are at home.

6.6.2 Optimisation Attempts for File Retrieval Predictions

AccessRank considers factors that are common to all domains to make its predictions. This allows it to be used in a wide range of contexts, however specialised algorithms have the potential to further improve accuracy in specific domains. Several factors specific to the file retrieval domain are described below, although they ultimately did not prove successful in improving prediction accuracy.

Path Components

Chapter 5 found considerable reuse within parts of the file hierarchy. For two paths, p_1 and p_2, let p_A be their deepest common ancestor, and $g(p_a, p_b)$ be the difference between the number of path components of p_a and p_b. Let $d(p_1, p_2)$ be the hierarchical distance between p_1 and p_2, calculated as follows:

$$d(p_1, p_2) = \max(g(p_1, p_A), g(p_2, p_A))$$

Let $d_i(p) = d(p, f_i)$, where f_i is the path accessed i retrieval events in the past, such that f_1 is the previously retrieved path. It is therefore conceivable that $d_i(p)$ affects the probability that path p will be accessed next, for low values of i – this would imply that accessing a file affects the chance that nearby files in the hierarchy will be accessed. However, simulations and analysis showed that it provided little predictive power, using either hard-coded weightings or weightings that self-adjusted based on access history, and considering either a range of i values or just $i = 1$.

Retrieval Method

The Places Frecency algorithm [143] considers the way a link is accessed in a web browser. For example, a link that is typed in the URL field will carry more weight than one that is clicked on a page. Similarly, files can be retrieved using different methods, such as using navigation or search.

Simulations using different weightings for different retrieval methods, even self-adjusting weightings, failed to improve accuracy. In fact, for the purposes of predicting navigation retrievals, accuracy was *higher* for a dataset that only included past navigation retrievals, compared to a dataset with all file retrievals that included dynamic method weightings. This suggests that there is little overlap between the sets of files retrieved using different retrieval methods. Furthermore, it suggests that when predicting retrievals with a particular method, retrievals made with other methods offer little, if any, predictive power, and that only considering retrievals made using the method in question may result in better accuracy.

Browser Windows

Chapter 5 found considerable window reuse in file browsers, with different users exhibiting different patterns of behaviour. When restricting predictions to just those performed within a file browser, the path locations of inactive browser windows (those other than the one being used for the current retrieval) could potentially influence which items are most likely to be accessed next. For example, users who exhibit *hoarder* behaviour often create new windows instead of reusing existing ones that have the potential to be useful in the future, and may end up with redundant windows that suggest that retrievals of items in their locations may be more likely. On the other hand, other users may use windows more efficiently, and retrievals for items stored in or nearby inactive browser window locations may be less likely, else those windows would have been used for the retrieval instead.

Paths were classed as one of four categories: (1) visible in an open, inactive window (the active window will always contain the retrieved file immediately before it is opened); (2) a descendent of an inactive window's location; (3) in a folder that is an ancestor of an inactive window's location; or (4) all

other cases. Simulations were performed using these categories as a factor when performing predictions, using self-adjusting weights to acknowledge the different behaviours exhibited by different classes of users. While the final weights did suggest that the locations of open windows had some correlation to retrieved files, and that there were large differences between participants, the amount of noise meant that the window locations offered little predictive power.

6.6.3 Applications

AccessRank could be incorporated into any user interface containing patterns of reuse. Some examples are:

- A file browser, to assist with the often tedious task of finding files. Sometimes files' exact locations are unknown, which makes finding them slow, and even when they are known, it can be a slow process getting to them if they are deep in the folder hierarchy. Search can help, but is unreliable and often only used as a last resort [19]. Figure 6.11 shows a mockup of AccessRank incorporated into a web browser-like path field at the top of a file browser. In the example, the user has typed 'so', and a list of recommendations of relevant files has appeared. This makes accessing common files as easy as typing in just a couple of characters. Additionally, AccessRank is incorporated in a second way, shown at the bottom left of the mockup. Suggestions are provided for files and folders that the user might be looking for, without using a hint. These suggestions are provided in a separate view, without affecting the core parts of the interface, an approach that previous research has suggested is effective [74]. These suggestions could be narrowed down by only suggesting items in the folder hierarchy subtree rooted at the current location. Other interfaces that utilise AccessRank to aid with file retrieval are introduced in Chapter 7 and extended in Chapter 8.

- While most modern command line interfaces allow users to acquire previous commands using the up and down arrow keys, they could be improved by incorporating AccessRank. Figure 6.12 shows one possible

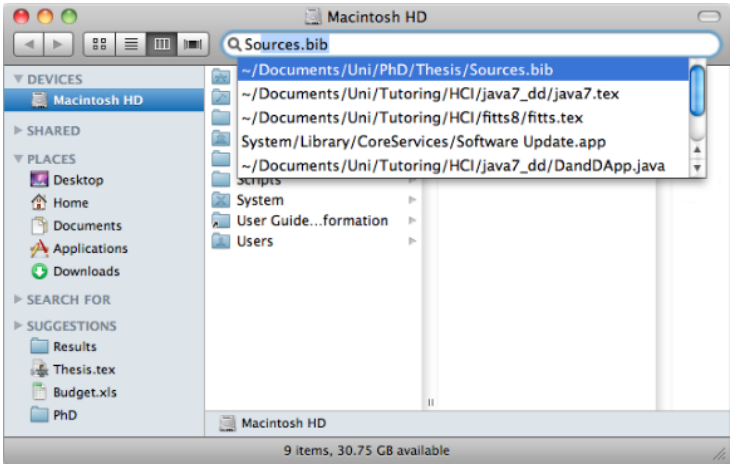


Figure 6.11: Mockup of a file browser, based on OS X's Finder, giving suggestions with a hint (top) and without (bottom left).

way of doing this, by displaying a drop-down list of suggestions as the user types. If the desired command is at the top of the list, the enter key would execute it, making common but long commands a lot simpler to enter. Domain specific improvements could be implemented, such as taking into account the current directory or by incorporating a Markov model based on just commands rather than the entire command line (for example, a 'javac' command to compile a Java application will often be followed by a 'java' command to run it).

- Web browsers could incorporate AccessRank to improve their rankings of suggestions when typing in the URL field. Domains specific factors such as those in the *Places Frecency* algorithm could also be incorporated into AccessRank.

- Expert users often have a lot of windows open at once. Most window switching interfaces do not cater for this, and distinguishing between windows can become difficult. While ordering windows using AccessRank could be confusing to people who are used to a predictable ordering, it could be of use to augment visualisations. For example, items which are more likely to be relevant could be highlighted in a brighter colour or have a larger preview.

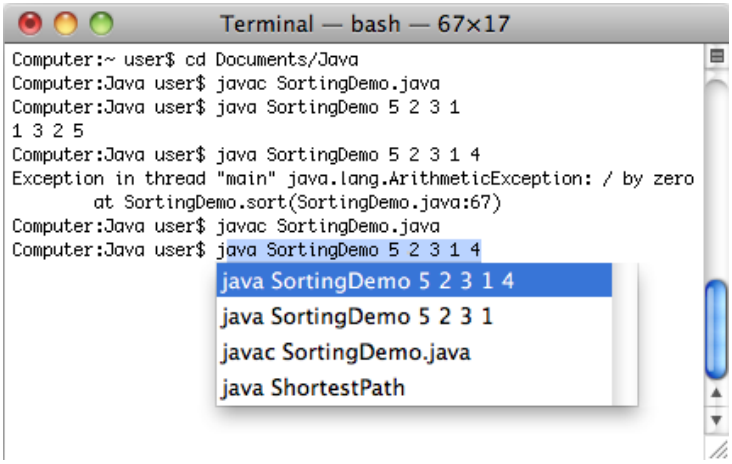


Figure 6.12: Mockup of a terminal giving suggestions.

- Elsweiler et al. [64] found that emails that have previously been revisited are significantly more likely to be revisited again. They argue that there could be considerable benefits to interfaces that predict email revisitation, and suggest that highlighting likely emails could greatly aid revisitation.

6.7 Conclusion

This chapter introduced AccessRank, a new prediction algorithm designed for user interfaces, with a focus on both predictive accuracy and the stability of prediction lists over time. AccessRank incorporates aspects of existing algorithms, with the addition of a stability component that restricts ranking changes. AccessRank was compared to existing algorithms by converting logs from a range of domains into a standardised format, and running simulations with each algorithm while measuring results with a range of accuracy and stability measures. Results show that AccessRank variations outperform existing algorithms when considering a combination of accuracy and stability, and that AccessRank is therefore well-suited to use in user interfaces.

Analysis also provided insights into the relative performance of algorithms in different domains, providing guidance for appropriate domain-specific algorithm selection. In particular, the benefits that AccessRank offers compared

to existing algorithms vary considerably based on the domain it is used in. In the domain of file retrievals – the focus of this thesis – recency was an important predictor, and the benefits that AccessRank offered were smaller than in some other domains. However, it still provided the best accuracy-stability tradeoff when appropriately configured. Further discussion about the use of AccessRank for file retrievals is discussed in later chapters; Chapter 7 describes new file retrieval techniques that incorporate AccessRank, and Chapter 8 describes real world implementations of these techniques which incorporate several specialised modifications to AccessRank.

Part IV

Improving File Retrieval

Chapter VII

Preliminary Design & Evaluation of Improved Navigation-Based File Retrieval Interfaces

The characterisation of file retrieval behaviour described in Chapter 5 confirmed previous results that navigation-based file retrieval is the preferred form of file retrieval [19, 25, 34]. By allowing users to locate files using recognition rather than recall, it has a lower cognitive load than other methods such as search [25, 176].

Despite users' preference towards it, results of the characterisation study showed that search often resulted in lower retrieval times than navigation-based techniques. Specifically, mean retrieval times were 5.7 s when searching in the Spotlight menu, but 10.2 s when using navigation. Bergman et al. [30] found similar navigation retrieval times of 12 seconds per retrieval for Mac users and more than 17 seconds for Windows users. This is a long time to retrieve a file, given that selecting a ready-to-hand file icon would take no more than a second or so. There are two main reasons for these long retrieval times: people may not know where a file is, resulting in extra time to explore the file system; and there are numerous navigation actions required (remembering folder names, finding icons in the current display, and clicking folders to open them). Regardless of the cause, it is clear that reducing retrieval time for hierarchical file browsers could result in large aggregate time savings.

The general problem of improving file retrieval has received substantial research attention. Researchers have studied several aspects of the problem: the ways that users choose to organise information [127, 29], the performance implications of different hierarchical structures [120], potential improvements to file access using search [53], and visualisations that provide shortcut access

to files [170]. Commercial systems have also iteratively refined their facilities, with tools such as "Open Recent" menus, full text searching, and aliases or shortcuts now standard in most operating systems.

Although some alternative retrieval techniques have been shown to be faster than standard navigation [170], the performance improvement is often the result of specialised interfaces that can not be used for all files (e.g., "Open Recent" menus). Other times, they come at the cost of switching to a completely different retrieval paradigm. This is a problem, due to people's continued preference for hierarchy-based file navigation over alternative methods. Bergman et al. [25] give four explanations for this preference: first, the locations and mechanisms of navigation-based retrieval remain consistent and reliable, whereas the organisation and content of search results can vary from one retrieval to the next; second, navigation reduces cognitive load because users can rely on recognition of the steps toward the target rather than needing to recall file attributes; third, the mental and physical mechanisms used for retrieval may become partially automated due to their consistency, allowing users to remain focused on their work; and fourth, the location-based mechanisms of hierarchical containment are familiar from the real world, which may serve an important sense-making function. In contrast, research has shown that search, although an essential tool for some retrievals, is used mainly as a method of "last resort" [145, 25], called upon when users cannot remember the location of files in the hierarchy.

These preferences imply that performance improvements to navigation-based file retrieval would be highly beneficial to users. To explore possible improvements, this chapter presents three new techniques that work within the existing presentation styles and interaction models of the standard hierarchical file browser. The new techniques are based on design goals for overcoming three performance constraints in navigation-based retrieval (described in greater detail later in the chapter):

1. Overcome the *visual search* constraint: minimise the time spent at each hierarchical level, by reducing exploration and visual search;

2. Overcome the *levels* constraint: reduce the number of levels traversed, by facilitating shortcuts;

3. Overcome the *practice* constraint: improve navigation expertise, by promoting rehearsal of the retrieval mechanics.

The three new techniques work by tightly integrating results into the interface and interaction paradigm of the file browser. To reduce step times during file navigation (goal 1), *Icon Highlights* (Figure 7.1a) predict which items in the current folder are most likely to be accessed, and give them greater visual prominence. *Hover Menus* (Figure 7.1b) provide quick access to commonly accessed items inside folders, in order to reduce the number of steps required in many cases (goal 2). *Search Directed Navigation* guides users through a file hierarchy based on a filename query. This is designed to bring some of the advantages of search to file navigation, while facilitating the development of expertise when compared to search (goal 3).

A two-part experiment validates the new techniques. The first part of the study examines user performance and preference with the techniques in comparison to standard file browsers, using tasks that involve retrieving both previously-visited and unvisited files; it also examines how well users learn file locations with the techniques. The second part of the study examines the effects of spatial stability of the folder contents on the relative performance of the techniques, which is important when folder content varies and when view modes change. The study's results show that the new techniques provide substantial performance improvements over a standard file browser, and that they are strongly preferred by users. The chapter concludes with a discussion of how the techniques could be combined into a single file browsing interface, future refinements, and limitations of the study.

7.1 The Performance Impact of Structure

The influence that hierarchical structure has on navigation time has been extensively researched, and is broadly encapsulated by the question "broad and shallow or narrow and deep?" [140, 120]. Cockburn and Gutwin [45] provide a review of thirty years of empirical research on the topic, which shows diverse results. Several studies show that plots of navigation time against depth follow a "U-shape", while others show that time increases with depth. Cockburn and Gutwin also present a simple mathematical performance model,

called "Search/Decision and Pointing" (SDP), that explains the result's diversity: broadly, a U-shape occurs if users must visually search for items at each level of the hierarchy, while shallow structures perform best when users can anticipate target locations at each level.

The SDP model inspired the design goals presented in this chapter, and so it is briefly summarised here. SDP predicts the time taken to select an item at one level of a hierarchy by combining three factors: the time to visually *search* for the target; the time to *decide* about target location; and the time to *point* to it. Pointing time is modelled using Fitts' Law [70]. Visual search is modelled as a calibrated linear function of the number of candidate items, and it is employed when users have no basis for anticipating a target's location (e.g., the user is a novice or the interface is unpredictable). Decision time is modelled as a calibrated logarithmic function of the number of candidate items, and is employed when users can anticipate target location (e.g., the user is experienced with a stable display, or the interface presents a predictable dataset, such as an alphabetic list of items). The model uses a power-law to predict the user's transition from search-based strategies to decision-based ones when interfaces offer consistent access methods across retrievals. Finally, the model sums the predicted time for each level across hierarchical levels.

The SDP model suggests three promising opportunities for improving performance in hierarchical navigation, detailed later: (1) reduce visual search time, which can be a performance bottleneck due to its linear function of candidate items; (2) reduce the number of hierarchical steps required through the hierarchy; and (3) help users transition from relatively slow search-based strategies to faster decision-based ones.

7.2 Improved File Navigation: Goals and Interfaces

Previous work suggests several opportunities for improving human performance in navigation-based file access. This section distills these opportunities into three design goals, and then presents three interfaces aiming to satisfy the goals. It is important to note that any file retrieval mechanism will have three parts: an underlying algorithm for deciding which items to

present, a presentation approach for making items visible and salient, and a set of interaction techniques with which the user can select items and specify targets. The primary focus is on the presentation aspects of the techniques, but some elements of the algorithms and interaction techniques will also be discussed as needed.

7.2.1 Design Goals

The design goals stem from insights within the "Search Decision and Pointing" model [45] which predicts hierarchical navigation time based on the time taken at each hierarchical level ("step duration"), the number of levels traversed ("step count"), and the potential for the user to make a transition from novice to expert behaviour.

Goal 1: Minimise time spent at each hierarchical level

The SDP model suggests three possibilities for reducing step duration, by improving human performance in (a) searching for targets, (b) deciding about them, and (c) pointing to them. Extensive prior literature has investigated improved pointing techniques ((c), see [15] for a review), and Goal 3 addresses the transition to decision-based methods (b). In Goal 1, therefore, the focus is on mechanisms that improve visual search (a). This is particularly important because visual search can be a performance bottleneck [45].

Reducing visual search time involves two activities. First, the primary job of the retrieval technique's underlying algorithm is to determine a small set of likely candidates for the current retrieval task. Second, the technique's presentation approach must make those candidates visible and salient, to reduce the number of objects that need to be visually inspected.

Prior work on methods for improving the presentation of candidates includes a variety of highlighting mechanisms that enhance the visual salience of likely targets without changing their layout, such as ephemeral adaptation [68]. Other techniques segment the visual presentation across space or time, by moving salient items into prominent positions (e.g., split menus [166]), by segmenting images (e.g., [71]), or by rapidly presenting images across time (e.g., [55, 71]).

Icon Highlights, described below, use AccessRank to predict probable items at each level of the hierarchy, and uses in-place highlighting to increase their visual salience.

Goal 2: Provide shortcuts to reduce levels traversed

The SDP model also suggests that efficiency gains can be achieved by reducing the number of hierarchical levels traversed en route to the target. Users might achieve this by creating flatter hierarchies or by creating shortcut links to important files and folders, but both require a-priori knowledge of the future need to retrieve particular files, as well as an understanding of the file structure's implications on retrieval times.

Hover Menus, described below, also uses AccessRank to determine a set of likely candidates. It creates separate candidate sets for each of the folders at the current level, thus indicating what is most likely within that folder's sub-hierarchy. The candidates are presented in a context menu associated with each folder, allowing users to skip one or more levels and get shortcut access directly to the most probable descendant targets in any branch of the sub-hierarchy.

Goal 3: Promote rehearsal to facilitate expertise

Kurtenbach [117] argues that expert performance is best facilitated when the actions that a novice uses for an interaction are a physical rehearsal of the mechanisms used when expert. Goal 3 adapts Kurtenbach's recommendation for file retrieval. While navigation-based retrieval naturally supports the goal by using consistent interaction mechanics across retrievals, search-based retrieval does not. Instead, search-based interfaces typically present their results to the user in a list, allowing direct access to each file. While this may facilitate rapid retrieval for the current item, the list presentation does not help users learn how to find the file in the navigation-based hierarchy – which is often the preferred mechanism of retrieval [19, 25, 32, 34].

Note that there is a tension between the recommendations of Goal 2 and Goal 3 because the presence of shortcuts allows alternative mechanisms for retrieval, and consequently it may reduce opportunity for rehearsal, particu-

larly if the shortcuts vary due to fluctuations in prediction consistency.

Search Directed Navigation, described below, allows users to type characters to determine the set of likely candidates based on filename matches. The presentation approach uses item highlighting (similar to Icon Highlights) to guide users through the hierarchy, with the intention of facilitating rehearsal-based transitions to expert performance.

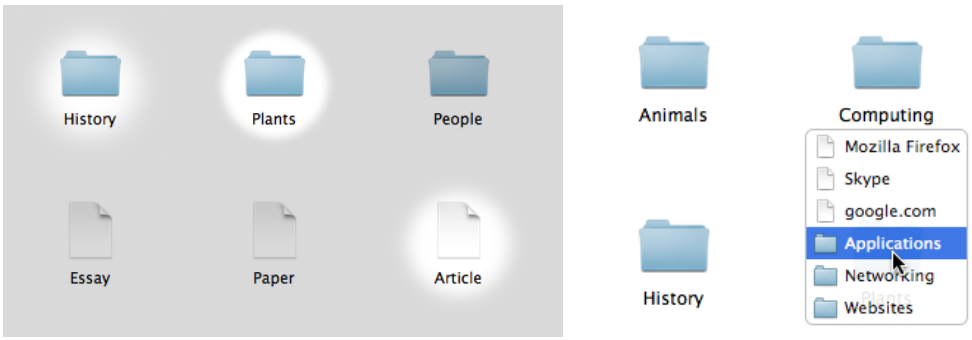
7.2.2 File Navigation Interfaces

A basic file browser was developed that closely mimics the icon view of the OS X Finder. This browser provided a framework in which to quickly investigate the feasibility of new interfaces without extensive development time, while also facilitating precise measurement of retrieval times by limiting the use of extraneous features. The browser was augmented with three interfaces that add new features intended to satisfy each of the design goals.

As described above, each technique can have an underlying algorithm, a presentation approach, and various interaction techniques. Two of the interfaces, *Icon Highlights* and *Hover Menus*, use AccessRank (Chapter 6) as their underlying algorithm to predict likely folders and files; both use the original version of AccessRank, with parameters $\lambda = 0.8$ and $\delta = 0.5$. *Search Directed Navigation* uses character-based filtering to determine candidates, and also keeps track of prior revisitations. It uses an explicit interaction technique – that is, users type characters to specify the search target. In contrast, the AccessRank-based techniques use implicit information about selection and navigation history, rather than user input about the target. The sections below provide details of the presentation approach for the three retrieval techniques.

Icon Highlights

Icon Highlights increase the visual salience of items that AccessRank predicts, as shown in Figure 7.1a. The highlighting mechanism is similar to that used in Apple's System Preferences application (Figure 7.2), with graduated blurring of 'spotlight' highlighting dependent on the item's AccessRank score (crisp presentation for probable items, blurred for less probable ones). In-



(a) Icon Highlights (b) Hover Menus

Figure 7.1: Icon Highlights (left), showing 'Plants' as the most likely AccessRank prediction. Hover Menu (right) showing predictions under 'Computing', with $n = 3$.

place highlighting (rather than spatial or temporal segmentation) is used because it promotes predictable retrieval mechanisms, in line with Goal 3. The highlighting scheme can be applied to any folder view, such as large or small icons/thumbnails, or filename lists. All items remain selectable, regardless of their highlight state. To indicate the presence or absence of highlighted items outside the current scroll view, the design marks items in the horizontal or vertical scrollbar [98].

Details of the highlighting and blurring algorithm are described as follows. A blur level is calculated for all highlighted items, where 0 corresponds to no blur and 1 responds to maximum blur. First, a minimum blur level b_{min} is calculated, based on the largest item score in the folder as a proportion of the total scores (p_{max}) as well as the total number of accesses (n) of items within the folder – see Equations 7.1 to 7.3.

$$b_{min_1} = \frac{2}{3}(1 - p_{max}) \tag{7.1}$$

$$b_{min_2} = \max(0.125(5 - n), 0) \tag{7.2}$$

$$b_{min} = b_{min_1} + b_{min_2} - b_{min_1} b_{min_2} \tag{7.3}$$

The result of this calculation is that higher blur levels are used, and thus a greater degree of uncertainty is implied, when (1) the top-rated item does

not represent a large proportion of all accesses in the folder, or (2) when there is insufficient history to be confident in the predictions.

Next, blur levels are calculated for the k highest ranked items (according to AccessRank) – thus limiting the amount of visual clutter. Note that because of AccessRank's switching threshold, requiring a minimum score difference before two items can swap ranks in order to increase selection stability, these items will not necessarily be the items with the highest scores. Blur levels are linearly translated from AccessRank scores such that the item with the highest AccessRank score has a blur level of b_{min} and an AccessRank score of 0 corresponds to a blur level of 1. Finally, these levels are rounded to one of l discrete blur levels, so that users are not distracted trying to distinguish subtle differences between highlight prominences. Icon Highlights were evaluated using $k = 5$ and $l = 4$.

Hover Menus

When retrieving a file, users often know where it is, but must go through the sometimes tedious process of traversing through a hierarchy to get to it. This issue can be diminished with features such as bookmarks, shortcuts or aliases, or by structuring hierarchies to be more shallow, however these all require explicit user action ahead of time and are limited in the number of locations they work for.

Hover Menus are designed to automatically give users shortcuts to predicted targets located deeper in the hierarchy. They offer a recognition-based shortcut for reducing the number of levels that must be traversed through standard navigation actions.

When a user hovers over a folder for 500ms a menu appears below it (Figure 7.1b) showing lower level content (folders and files at any sublevel) that have the highest AccessRank ranks. Selecting a menu item navigates directly to the associated folder or file. To facilitate rapid browsing of several folders, menus appear immediately if the cursor enters a folder within 500ms of leaving a folder with a menu already displayed. Menus also fade out if the cursor leaves a folder without a selection being made.

The menus are populated with up to n files followed by n folders, each

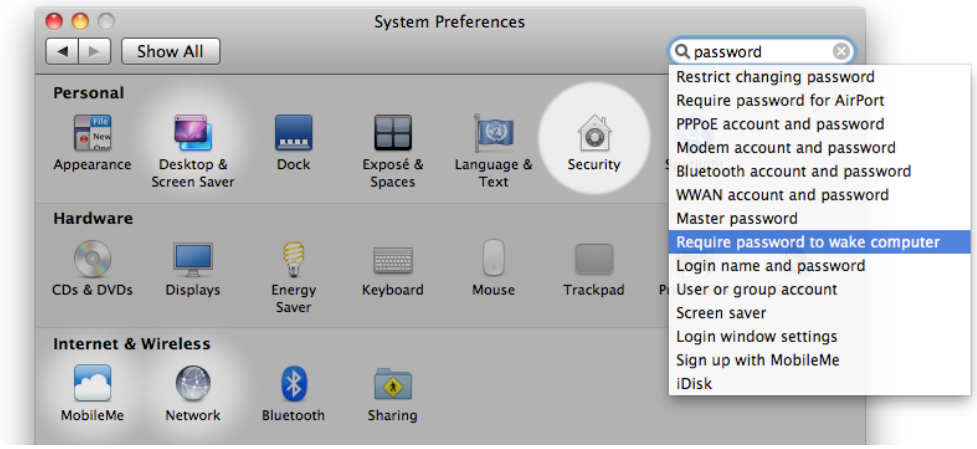


Figure 7.2: Search-based highlighting in OS X System Preferences.

sorted by AccessRank score. Hover Menus were evaluated with $n = 3$, although this value would likely be user-configurable in a deployed implementation.

This mixture of files and folders is useful in that it provides quick access to the files most commonly accessed, however it also provides a way to traverse partway through the hierarchy by selecting an intermediate folder when the target file is not listed.

Both Hover Menus and Icon Highlights are designed in line with Chapter 5's *aid revisitation, but only in context* design recommendation, as they facilitate revisitation without disorienting users by affecting the underlying presentation of the file hierarchy.

Search Directed Navigation

Search Directed Navigation (SDN) is intended to satisfy Goal 3 by providing search-based guidance through the navigation hierarchy towards the target, allowing the user's interaction with search results to be a rehearsal of navigation-based retrieval. It is in part inspired by a similar technique in OS X's System Preferences application (Figure 7.2), which highlights the icons for preference panes that contain a setting matching a search query.

SDN provides search-based guidance by highlighting items in the hierarchy that match typed query terms (Figure 7.3), using a visually similar

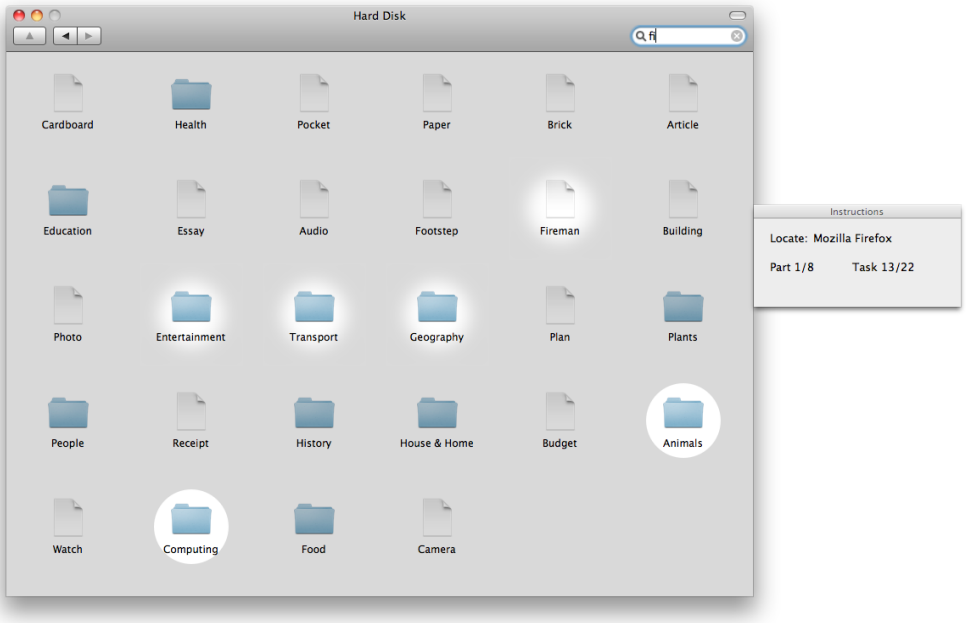


Figure 7.3: Experiment setup showing Search Directed Navigation. Predicted items (files and folders leading to files) are highlighted in response to search terms.

highlighting strategy to Icon Highlights. If the search term matches an item at any level, the highlighting propagates up the hierarchy to mark the path to the target – for example, if the query was 'budget', a folder named 'Finances' might be highlighted if it were to contain a matching item named '2013 Budget.xls'. An item matches the search query if any word (or consecutive combination of words) from the filename starts with the query. In the above example, the query '2013 bu' would match '2013 Budget.xls', but '13' and 'budget 2013' would not. In the current version, words are delimited by spaces, however a deployed version could also delimit by underscores, periods, hyphens or use of camel case (for example, MeetingMinutesNew could be parsed as three words), all of which are commonly used in filenames (see Chapter 5).

Items can be highlighted using one of several approaches. The evaluated implementation uses two highlight levels: one for previously-visited items (crisp border), and one for unvisited items (blurred border). Alternatively, items could be highlighted based on their AccessRank scores or through

other metrics. Full content-based search strategies could also be used to determine the set of matches, although the current implementation does not do so; this is because filenames are the most common attribute to search, as found in Chapter 5, and because restricting searches to filenames increases responsiveness and results specificity, in line with the *focus on attributes with high specificity* design recommendation from Chapter 5.

SDN is designed to address several deficiencies of search. Search is primarily used to retrieve files with unknown locations [25], however locations are more quickly learned when the easiest strategy available explicitly requires retrieval of location knowledge [63]. By merely aiding users to navigate through their hierarchies, SDN therefore assists users to rehearse, and thus learn, item locations, so that they can more easily transition to faster or more preferred methods (see the *facilitate location learning* design recommendation from Chapter 5). In contrast, other search interfaces typically only present a list of results, and do not have this properly.

Second, SDN is better suited than search for locating items with similar names where the item's location is the distinguishing element (see the *facilitate differentiation of items with similar names* design recommendation from Chapter 5). For example, one file called 'Presentation.ppt' might be located in a 'CHI 2013' folder, while another with the same name might be located in a 'Department seminar' folder. Clearly the folder name can immediately distinguish these two folders, however such information is rarely present in conventional search result interfaces.

Finally, while search as a method to retrieve files is typically used as a last resort due to the higher cognitive load of recall as opposed to recognition, the increased ability to use recognition with SDN means that users are potentially more likely to use it compared to search [19, 18, 32, 145].

7.3 Interface Evaluation

Icon Highlights (*IH*), Hover Menus (*HM*), and Search Directed Highlights (*SDN*) are each intended to improve different aspects of navigation-based file retrieval, as emphasised by their associated design goals: improve visual identification of likely targets, provide shortcut target acquisition, and im-

prove learning of file locations. If deployed in real file browser applications, these techniques are probably best combined (discussed later). However, it is important to first understand how each of the methods is used in isolation – how they affect file navigation performance, and how participants subjectively respond to them.

Devising experimental tasks for evaluating file retrieval interfaces is complicated by the number of relevant factors that might be considered. These include the structure of the file hierarchy, the degree of ambiguity in the semantic relationship between hierarchical components, the stability or predictability of the data organisation in the interface, the recency and frequency of any previous visits to items in the hierarchy, as well as individual user preferences. This led to an experiment design which uses artificial tasks that control, or partially control, some of these factors in order to yield preliminary insights into the comparative usability and performance of the interfaces (more extensive evaluation is described in Chapter 8).

Experimental tasks involved navigating within a three-level semantically organised hierarchy to select a target file that was cued by displaying its name in a small window at the side of the file browser interface. Example targets include 'Paris' (within 'Geography' then 'Capital cities') and 'Horse' (within 'Animals' then 'Mammals'). The path to most targets was intentionally partially ambiguous – for example, the target 'Paris' might reasonably be contained within top-level folders 'Geography' or 'History', or in second-level folders 'Capital cities' or 'Large cities'. The ambiguity was intended to emulate imprecise recall of file locations and imperfect organisation as is typical of personal file hierarchies.

During the experiment participants visited files up to 5 times each, allowing analysis of how performance with the interfaces changed as users become more familiar with the file locations and retrieval mechanisms. The experiment also examined how well users remembered the location of target files.

7.3.1 Participants and Apparatus

16 volunteers (9 female, mean age 25.2) participated in the study, which lasted 60-90 minutes. All had normal or corrected to normal vision and were

fluent English speakers.

The experiment ran on an iMac with a display resolution of 1920×1080. The file browser window was 880×631 pixels, populated with 48×48 pixel icons representing files and folders. Software logged all user actions, including task time.

The experimental file browser application imitated the icon view in OS X at a fixed size of 6×5 icons (Figure 7.3). Items could be opened by double clicking, and toolbar navigation links allowed for navigation back, forward, or up to the parent directory. The basic interface, called *Standard* (ST) was then augmented with each of the three interfaces introduced in this chapter. Icon Highlights (IH) used $k = 5$ and $l = 4$, Hover Menus (HM) used $n = 3$, and Search Directed Navigation (SDN) used two highlight levels for visited (crisp) and unvisited (blurred) targets.

The same file hierarchy was used throughout the experiment, with targets located in different branches for each interface. The hierarchy structure was modelled on Bergman's findings of mean folder sizes [29]: 12, 10 and 8 folders, and 16, 12 and 9 files at the root, second and third level respectively. Folders containing no target nodes were not populated. For consistency, all target files used for analysis were located at the third level, which matches the observed mean file retrieval depth in previous studies [29, 30]. In total, the hierarchy contained 388 folders and 448 files, of which 288 could be selected as target items.

During the experiment, a small information window provided participants with brief instructions, a progress meter, and (during tasks) a target filename. Information about where in the file hierarchy the file was located was not provided and had to be deduced based on the semantic structure. Opening the target file would return the participant to the root folder and begin the next task. Opening any other file would not do anything.

7.3.2 Procedure

The experiment consisted of two parts: the first asked participants to carry out retrieval tasks (using each of *ST*, *IH*, *HM* and *SDN*) with icons that remained spatially stable in the file browser (though initially ordered ran-

domly); the second repeated the procedure of part one, but used icons that were randomly rearranged after each selection in order to examine how performance with the interfaces was affected by unstable locations. Understanding susceptibility to unstable locations is important since many activities cause view instability, including changing a browser window's sorting criteria, view settings or size, or changing folder contents.

Part one of the experiment consisted of three phases using each of the four interfaces: *practice*, *retrieval*, and *standard-retrieval*. All three phases were completed using one interface before moving on to the next.

The *practice* phase allowed participants to familiarise themselves with the interface. The phase used a small training file hierarchy, and participants were instructed to open files freely until they were ready to start the actual tasks. All data from the practice phase was discarded.

During the *retrieval* phase, participants completed 22 file retrieval tasks in response to cued target filename stimuli. Successfully completing a retrieval automatically initiated the next by displaying another target filename. Target files were randomly selected according to a near-Zipfian distribution at each level of the hierarchy, with ten target files occurring with frequencies 5, 3, 2, 2, 2, 2, 1, 1, 1, and 1 times each. The remaining two retrievals were for two 'distractor' targets, which were each accessed once. The two distractor targets were at the top level of the hierarchy, and were included to reduce participants' anticipation of targets always being located at the third level. They were inserted roughly one-quarter and two-thirds of the way through the overall sequence of tasks. The sequence of other retrievals was randomised based on the above distribution of frequencies, but was the same for each interface for a given participant.

The ten true targets for each interface were randomly selected from the hierarchies contained within three top-level folders. To reduce learning effects stemming from familiarity with the hierarchy, different interfaces used targets from different top-level folders (thus using all 12 top-level folders over the course of the experiment). Note that the two interfaces that used AccessRank (IH and HM) had no pre-populated history of file accesses, so for the initial selections of each target they provided no automatic highlighting (*IH*) and no menus (*HM*). As the retrieval phase progressed, however, these systems

adapted to the user's navigation actions as described above. Because of the short duration of the experiment, a modified version of AccessRank was used that did not incorporate the time of day.

Once the *retrieval* phase was complete with each interface, participants completed NASA Task Load Index (TLX) worksheets [89] and provided comments on the interface (reproduced in Appendix B).

The *standard-retrieval* phase consisted of a single selection of each of the ten targets from the *retrieval* phase using the standard file browser (i.e., the IH, HM, or SDN augmentations were unavailable). Its purpose was to analyse any differences in how the interfaces supported users in learning the traditional mechanisms for navigating to files.

Part two of the experiment repeated the method used for part one, but without the *standard-retrieval* phase and using maximally unstable icons – the spatial location of every icon in each folder was randomised after each target acquisition. The targets used with each interface were randomly selected within the same top-level folders used for that interface in part one, so participants could be expected to have some familiarity with them from the beginning of part two. However, as with part one, the interfaces began with no usage history, and so provided no highlights or menus for the initial selections.

The following pseudo-code summarises the procedure:

foreach Part $\in$ {*stable, unstable*}
  foreach Interface $\in$ {*ST, IH, HM, SDN (counterbalanced)*}
    phase1: practice
    phase2: retrievals
    if Part = stable
      phase3: standard-retrievals

7.3.3 Experimental Design

Data from the *retrieval* phases are analysed using a 4×5 repeated-measures analysis of variance (ANOVA) for within-subjects factors *interface* (levels *ST, IH, HM, and SDN*) and *repetition* (levels $1, 2, 3, 4, 5$, representing the

count of repeated access to the same item). The primary dependent variable is total time to select the target file (log transformed to reduce the impact of positive skew). Navigational error rates are also analysed. Timing data from the *standard-retrieval* phase are analysed using one-way ANOVA across levels of *interface*, since each target is only visited once. To help characterise performance with the interfaces, step time (the time spent navigating down a level of the hierarchy) is analysed as a secondary dependent measure using factors *interface* and *depth* (hierarchical level $1, 2, 3$).

To reduce the impact of outliers when navigational errors were made, task times were capped at 30 seconds and step times at 10 seconds. This affected 3.4% of tasks and 1.7% of steps, distributed evenly between interfaces. Data from the first task for each interface in each phase was excluded to allow participants to get accustomed to each configuration. Post hoc tests use the Bonferroni correction. Where the ANOVA assumption of sphericity was violated (Mauchly test), Greenhouse-Geisser adjustments were used (as indicated by non-integral degrees of freedom).

7.4 Results

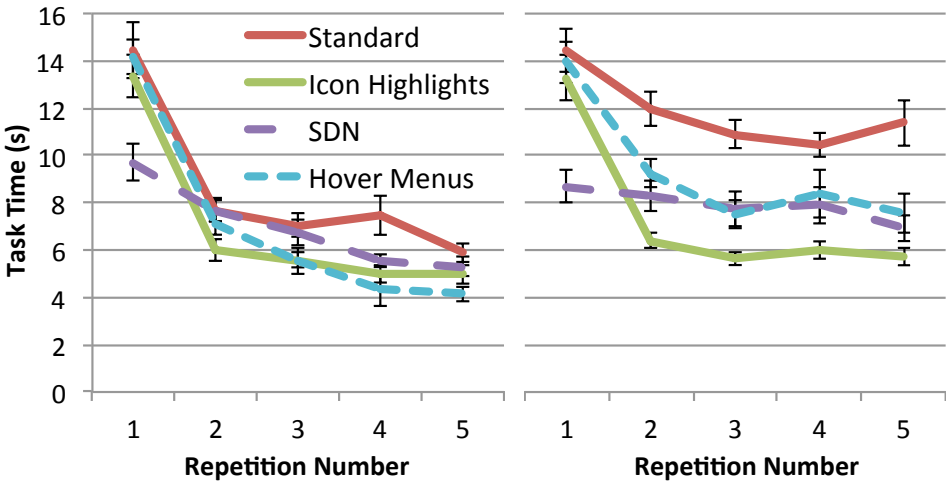
Results from part one (stable icons) are presented first, then part two (unstable icons), followed by subjective responses and further characterisation of how the interfaces were used.

7.4.1 Part 1: Spatially Stable Icons

All three interfaces improved performance over the standard file browser, with SDN providing the largest benefits for unvisited files, and IH and HM working best for revisitations.

Retrieval Times

Figure 7.4a summarises retrieval times. ANOVA showed a significant main effect of *interface* ($F_{1.9,28.8} = 8.2, p < .01$), with SDN, IH, and HM all similarly fast (means 6.98, 6.98 and 7.06 seconds respectively) compared with ST slower at 8.51 s. Posthoc analysis confirmed pairwise differences between ST and both SDN and IH.

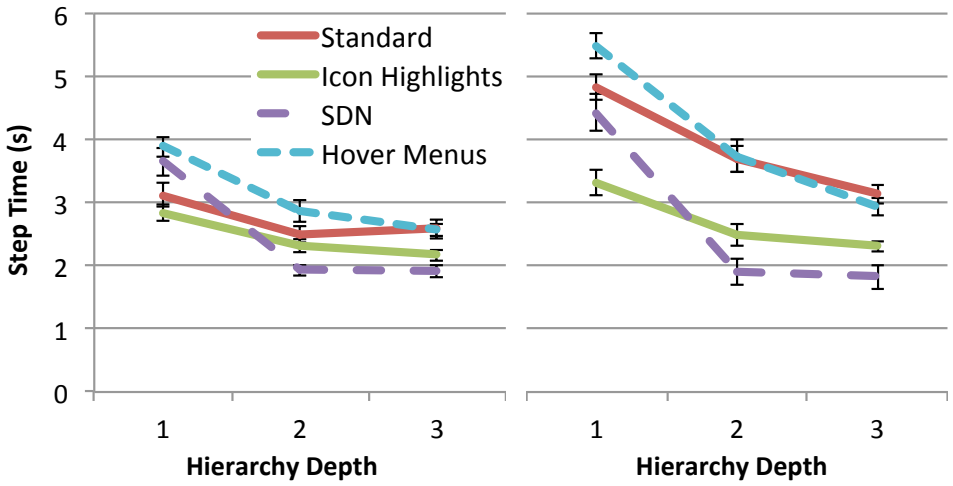


(a) Spatially stable icons (b) Maximally unstable icons

Figure 7.4: Task times by repetition number. Error bars ± 1 st. err.

Note that ANOVA results in the mean time for each repetition number being weighted equally when determining overall means, even though there were fewer tasks for higher repetition numbers (due to the Zipfian-like distribution of targets). This means that compared to retrieval times as a whole, those with high repetition numbers were overrepresented in ANOVA means compared to items that were only visited once. Unweighted means may therefore be better predictors of real world performance, where retrieval patterns similar to Zipfian distributions are typical. For these, SDN was fastest overall (mean 8.3s, s.d. 2.0s), followed by Icon Highlights (mean 9.3s, s.d. 2.1), Hover Menus (mean 10.0s, s.d. 1.8) and the standard icon view (mean 10.7s, s.d. 2.9).

There was an expected significant main effect of *repetition* ($F_{2.6,39.0} = 116.9, p < .001$), with mean times reducing from 12.9 s to 5.07 s for repetition 1 to 5. Of particular note, there was a significant *interface*×*repetition* interaction ($F_{7.1,106.8} = 7.1, p < .001$). Figure 7.4a suggests that this is best attributed to SDN being substantially faster than the other interfaces for the first retrieval of an item, but slower than IH and HM for revisitations. This is primarily due to differences in the techniques' underlying algorithms and interaction mechanisms, rather than their presentation approaches: SDN al-



(a) Spatially stable icons (b) Maximally unstable icons

Figure 7.5: Step times by depth in the hierarchy. Error bars ± 1 st. err.

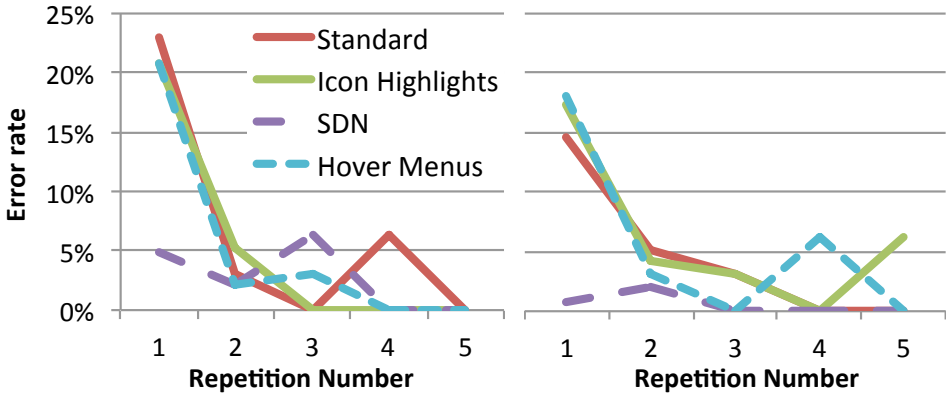
lows user input, and so can work immediately, whereas IH and HM must build up information about user selections before they can make good predictions.

To summarise, the standard interface (ST) was consistently slowest. Search Directed Navigation (SDN) was fastest for the first retrieval, when users had not yet learned item locations (and when the other interfaces had no history to work on), but third fastest with repeated items. Hover Menus (HM) were relatively slow for initial selections, but fastest once their menus had been populated with shortcuts. Its fast performance can be explained by users quickly finding the top-level folder, and capitalising on the menu shortcut to the target item. Icon Highlights (IH) was first or second fastest at each repetition level.

Step Times During Retrieval Phase

A step time was defined as the time taken at each hierarchical level (i.e., a 'step') from a user's final arrival at a particular level (through a previous selection or feature such as the 'back' button) until their final departure. Note that the sum of step times for a task may be less than the total task time, since superfluous navigations are not included in any step time.

Figure 7.5a shows step times for each interface across file hierarchy depth.



(a) Spatially stable icons (b) Maximally unstable icons

Figure 7.6: Error rates by repetition number

There were significant main effect for both *interface* and *depth* ($p < .001$). Overall step times were lowest for Icon Highlights and SDN (mean 2.6s, s.d. 0.6 and 1.4 respectively) followed by the standard icon view (mean 3.0s, s.d. 0.8) and Hover Menus (mean 3.7s, s.d. 1.1). Post hoc comparisons showed all differences were significant ($p < .001$) except for SDN and Icon Highlights. The advantage that SDN had for task times was reduced for step times as they do not incorporate SDN's lower error rate (see below).

There was also a significant *interface*×*depth* interaction ($p < .001$). SDN and HM both showed particularly slow performance at the first level, which can be attributed to their interaction mechanisms – the time costs of typing a query (SDN) or browsing menus (HM). However, SDN provides faster times at deeper levels through specific item highlighting, and HM reduces the number of steps required to retrieve a file.

Error Rates During Retrieval Phase

Any navigation into an incorrect folder was logged as an error. However, as stated previously, the hierarchy was intentionally partially ambiguous, so high levels of error were anticipated, particularly for the first selection of any target.

Figure 7.6a summarises the results, showing no significant effect of *interface* ($F_{3,45} = 1.8, p = .16$), but an anticipated effect of *repetition* ($F_{2.6,38.8} =$

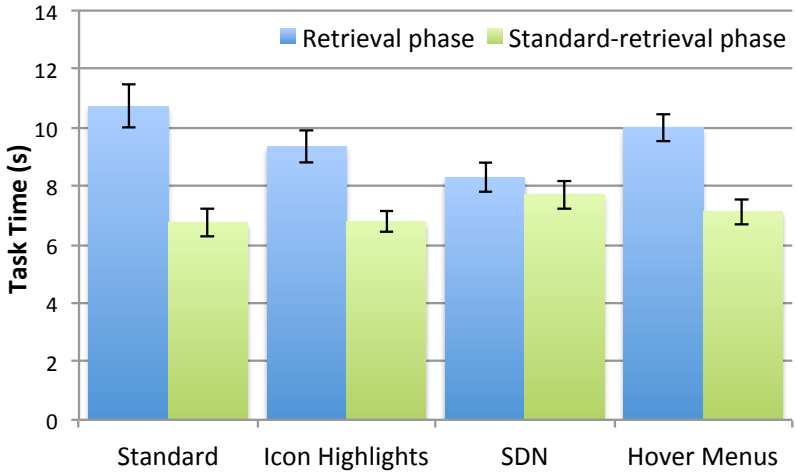


Figure 7.7: Task times for retrieval and follow up standard-retrieval phases after removing the augmentations. Error bars ± 1 st. err.

33.7, $p < .001$). A significant *interface*×*repetition* interaction ($F_{4.4,66.0} = 3.15, p < .05$) is best attributed to the marked difference between SDN's low error rate in the first repetition (4.9%) compared to other techniques (20-23%). This is in part an artefact of the experimental method, which used exact filename stimuli (favouring SDN) but inexact, ambiguous hierarchical structures to guide navigation-based retrieval.

Standard Retrieval Phase

The standard retrieval phase involved navigating to targets without any of the augmentations provided by IH, HM or SDN, to test for differences in participants' learning of the actual file locations. There was no main effect for *interface* (p = .338), with mean times ranging from 6.8 s with ST and IH to 7.7 s for SDN, shown in Figure 7.7. Error analysis also showed no significant effect.

7.4.2 Part 2: Maximally Unstable Icons

Results from part 2 showed that the interfaces provide additional value when icon locations are not spatially stable.

Retrieval Times, Step Times, and Error Rates

Figure 7.4b summarises the results for retrieval times, showing that the new interfaces yield greater benefits with unstable icons than with stable icons. Mean retrieval times were fastest with IH (6.1 s, s.d. 1.0), followed by SDN (8.0 s, s.d. 2.5) and HM (8.6 s, s.d. 2.1), with ST much slower at 11.5 s (s.d. 2.3) ($F_{1.9,28.7} = 25.4, p < .001$). Posthoc analyses show pairwise differences between ST and all three augmented interfaces, as well as between IH and HM. *Repetition* showed the anticipated significant main effect ($p < .001$). As with spatially stable icons, unweighted means resulted in SDN performing better relative to the other interfaces as initial visits (where it performs best) were no longer underrepresented; mean retrieval times with this method had SDN fastest (8.3 s, s.d. 2.6), followed by IH (9.5 s, s.d. 2.2), HM (11.2 s, s.d. 2.4) and ST (12.9 s, s.d. 2.8).

The significant *interface*×*repetition* interaction ($F_{5.5,82.2} = 6.4, p < .001$) is again attributed to SDN showing much less performance improvement across repetition compared with other interfaces. The figure also shows that IH performed particularly well with unstable icons when revisiting items.

Analysis of step times showed similar (though larger) effects to those described above for stable icons (see Figure 7.5b), with both main effects and the interaction significant ($p < .001$). Overall step times were again lowest for IH and SDN (mean 2.7 s, s.d. 0.8 and 1.5 respectively) followed by ST (mean 3.9 s, s.d. 1.0) and HM (mean 4.1 s, s.d. 1.3). Post hoc comparisons showed all differences were significant (p < .001) except for between SDN and IH, and between HM and ST.

In analysing errors, there were significant main effects for interface (p < .05) and repetition (p < .001), as shown in Figure 7.6b. SDN had substantially fewer errors (0.6%) than the other interfaces (4.6, 6.2, and 5.5% for ST, IH and HM respectively), with consistently low error rates, unlike the other interfaces, which had much higher error rates for the first repetition. Post hoc comparisons confirmed that SDN's advantage was significant, but this can again be partly attributed to the experimental method.

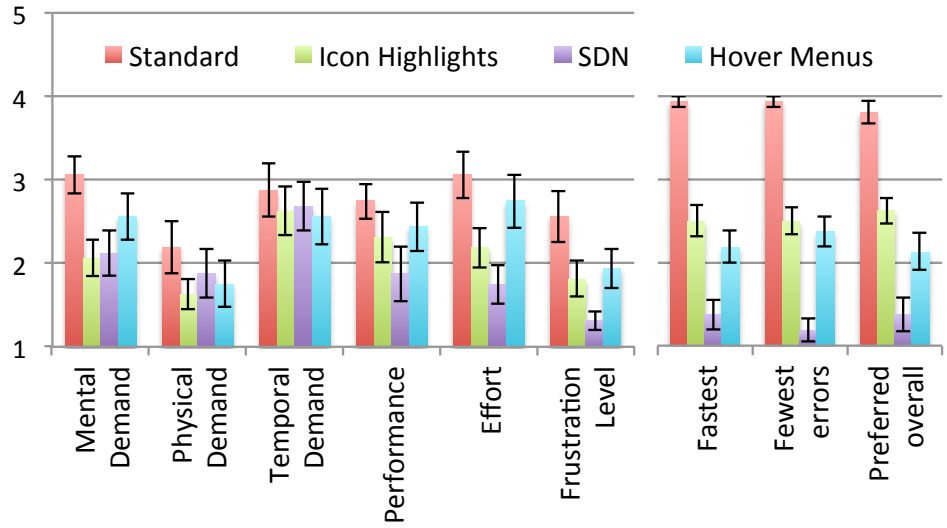


Figure 7.8: NASA TLX scores (left) and subjective rankings (right). Lower is better. Error bars ± 1 st. err.

7.4.3 Subjective Results

Subjective responses in all categories of the NASA TLX worksheets favoured the new interfaces in comparison to the standard file browser. Friedman tests show significant differences between interfaces for mental demand, performance, effort and frustration. Participants also ranked the interfaces (1st to 4th) for speed, errors, and overall preference. SDN was consistently ranked first, and the standard interface consistently last. Responses are summarised in Figure 7.8.

When asked whether they would prefer SDN or traditional file search tools, 12 of the 16 chose SDN. However, several participants commented that directly listing results was an advantage of search, with one noting that SDN would be a good complement to search. One noted that she would like a combination of HM and SDN as a way to get the advantages of SDN's highlighting but with fewer navigation steps. Another noted that SDN was most useful when first accessing items, but after a few times they remembered their locations, as intended in design goal 3.

Two participants commented that the Hover Menus appeared too slowly, suggesting the need for a shorter dwell timeout or the ability to immediately show the menu (e.g., using a mechanism similar to right click).

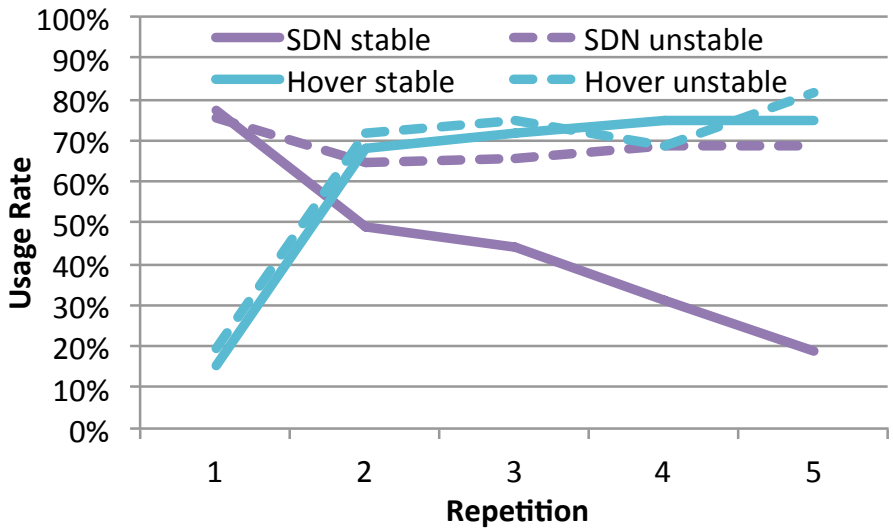


Figure 7.9: Percentage of tasks where SDN and Hover Menus were used, by repetition number.

One participant also commented that Icon Highlights made it harder to select unhighlighted items, because the highlighting dragged their eyes to items. Another complained that it highlighted folders that were mistakenly opened. This problem could be addressed by only updating the visitation data for a folder when a leaf node it contains is ultimately selected.

7.4.4 Characterisation of Use

Search Directed Navigation and Hover Menus require active use – users must explicitly choose to type a query or use a menu. Interaction logs revealed details about how participants used these interfaces.

Figure 7.9 shows the proportion of SDN and HM tasks in which their features were actively used, across repetition of access. The interesting result is in the contrast between SDN usage patterns when navigating stable and unstable icons. With stable icons, participants made extensive use (77%) of SDN in the first repetition, decreasing to 19% in the fifth repetition, suggesting that they gradually learned icon locations and decided not to use explicitly typed queries as they could quickly select items at their known locations. With unstable locations, however, participants continued to use queries across repetition (65-76% of tasks for all repetitions). Referring back

to the Search, Decision and Pointing model that motivated design goal 3, this suggests that users will naturally transition to decision-based mechanisms for identifying and selecting items when stable spatial locations allow them to do so, but that they prefer to use explicit search criteria instead of completing the time-consuming activity of visually searching for targets when items appear in random (unlearnable) locations.

Hover Menus were used consistently regardless of the stability of icons. Only the first repetition, where the menus were unlikely to be populated with assistive data, showed low levels of use. When menus were shown for stable icons, target items were present 58.0% of the time, and their parent folders 80.0% of the time, including 62.3% of the time when the target file was not. When the target file was present, it was selected 99.0% of the time; when absent, but with its parent folder present, the parent was selected 93.0% of the time.

For unstable locations, the target item's parent folder was much more likely to be present: 87.6% of the time (compared to 80.0%), and 79.1% of the time when the target file was not present (compared to 62.3%). This was partly because revisitations made up a slightly larger proportion of menu activations, and may have also been because participants more actively used them to revisit folders, as visual search times were higher. In fact, parent folders were selected in 98.1% of cases when the target item was not present in a Hover Menu but its parent folder was.

Overall, Hover Menus were activated for 54.0% of tasks in the hover menu condition, including 72.5% of revisitations. A selection was made in 81.7% of tasks where Hover Menus were activated.

7.5 Discussion and Future Work

Results show that all three interfaces improve performance in navigation-based file retrieval over the standard icon view. Icon Highlights and Hover Menus are particularly efficient for revisiting items, while Search Directed Navigation performs best for newly (or rarely) visited items. Furthermore, analysis of step times suggests that SDN would have higher relative performance for targets deep in the hierarchy, and the same is likely true for Hover

Menus as deep targets present opportunities to skip a larger number of levels than could be skipped in the evaluation.

An important finding was that, relative to the standard icon view, the three interfaces' performance increases as spatial stability decreases. Both IH and SDN partially overcome spatial instability by quickly focusing attention to relevant icons. Hover Menus still requires visual search to find a target folder, but the shortcuts it provides eliminate the need for visual search to be repeated at every level of the hierarchy. The standard view was notably slow with unstable icons, important because item locations can change when folder content changes or when window settings or sizes are changed. As file browsers have some degree of, but not complete, spacial stability, the magnitude of the performance benefit that the augmentations offer is likely somewhere between that shown in the two graphs in Figure 7.4.

7.5.1 Combining the Interfaces

As anticipated from the design goals, each of the interfaces performs best in different scenarios of use. For example, the Icon Highlights presentation of AccessRank predictions was the best interface for frequently-revisited files in spatially-unstable views, and SDN was best for infrequently-accessed files that are deeper in the hierarchy. Importantly, however, the interface designs are complementary and can be combined to gain increased benefit. In such a combination, Icon Highlights would be always present, while the features of Hover Menus and SDN would be available to the user on-demand.

Icon Highlights and SDN currently use the same visualisation to highlight items. This overlap is unlikely to cause confusion, since users who enter a search query to SDN have made an explicit choice to use that technique, and will likely expect that the highlighting is reflecting their search rather than the general AccessRank prediction. However, one of the augmentations could potentially use a different visualisation to make it easier to distinguish modalities, or to support both visualisations simultaneously. This could be achieved through use of colour, outlines, or any number of other approaches. One such alternative is presented for Icon Highlights in Chapter 8.

Hover Menus could be modified to adapt to SDN queries. Ultimately

this combination converges with traditional lists of search results, but could be implemented to either highlight matching menu items or repopulate the menu accordingly.

One area for further study in combining these interfaces is to determine whether users face additional cognitive load in deciding which technique to use. In a deployed interface that combined the techniques, users may default to using the automatic feedback provided by Icon Highlights, and only resort to Hover Menus or SDN when retrieval difficulties arise. This seems similar to current retrieval (where users resort to search after fruitless navigation), but the combined interface would provide a richer set of tools for assisting navigation without a complete change in retrieval mechanism.

Table 7.1 summarises properties of all the retrieval methods discussed in this chapter, including navigation, search, the three augmentations and a combined interface. Values are based on a mixture of evaluation results and theoretical underpinnings. While all approaches have strengths and weaknesses, in theory, combining IH, HM and SDN provides the most benefits.

	ST	**Srch**	**IH**	**HM**	**SDN**	**Cmb**
Reduces step times	✗	N/A	✓	✗	✓	✓
Good for deep targets	✗	✓	✗	✓	✓	✓
Location learning	✓	✗	✓	✗	✓	±
Copes with instability	✗	✓	✓	±	✓	✓
Reminding ability	✓	✗	✓	✓	✓	✓
Fast revisitations	✗	✗	✓	✓	✗	✓
Handles unknown locations	✗	✓	✗	✗	✓	✓

Table 7.1: Properties of retrieval methods discussed in this chapter (Srch = Search, Cmb = Combined). ✓= yes, ✗= no, ± = mixed support.

7.5.2 Interface Refinements and Implementation

Implementing the current designs of Icon Highlights, Hover Menus and SDN requires relatively minor cosmetic changes to the input and output behaviour of current file browsers – highlighting probable items, adding marks in the scroll-trough to demark their location, adding hover menus, and adding a search query field. Implementing the back-end predictive algorithms requires

recording and tracking file access data, as well as efficient algorithm implementations that can handle mutable hierarchical data; example implementations are described in Chapter 8.

The current implementations of IH, HM and SDN are relatively rudimentary, developed to explore their key concepts. There are therefore several avenues for interface refinement, including adapting them to other view types, and enhancing Hover Menus to provide more information about file and folder locations (e.g., by displaying full paths in tooltips).

In the evaluation, all folders that appeared in Hover Menus were at the same depth in the hierarchy, and file scores and folder scores were not directly compared. This meant that the underlying algorithms did not have to consider how scores of folders at different levels in the hierarchy should be compared, when higher level folders that encompass others are likely to be accessed more often, while lower level folders allow users to skip a larger portion of the hierarchy. This issue is discussed further in Chapter 8.

Search Directed Navigation performed best relative to a standard file browser when retrieving items for the first time. However, such items are displayed less prominently by the current implementation of the technique. This may indicate that alternate methods of determining highlight prominence may be more appropriate.

7.5.3 Limitations

The initial evaluation of the three interfaces provided important first results to investigate their feasibility. However, there were several limitations of both the design of the techniques themselves, and the experimental method used to evaluate them. Some of these are resolved in Chapter 8, however they are listed here to clarify the extent to which the initial results can be generalised.

Limitations of the Techniques

Highlights: The highlighting method is based on the 'spotlights' seen in OS X's System Preferences (Figure 7.2), but other methods may further improve performance. For example, Findlater et al. [68] described ephemeral

adaptation, where predicted items appear immediately, while other items gradually fade in. They found that ephemeral adaptation was more effective than static highlights, with both subjectively preferred to a control condition. While ephemeral adaptation may further improve performance for IH and SDN, its advantages come from its initial animation, after which predicted items are no longer distinguishable from remaining items. This results in several limitations for use in file retrieval: notably, it cannot be used with occluded highlights (i.e., where scrolling is required), where the animation would have concluded before the item is in view, nor does it support later reuse of a window, which, as found in Chapter 5, is extremely common in the domain of file retrieval. Static highlights – which are always visible – retain their benefits when scrolling or reusing windows. Further work is required to investigate whether the advantages of ephemeral adaptation outweigh these limitations, or whether other highlighting methods would further improve the interfaces described in this chapter.

AccessRank: The results do not indicate whether AccessRank, the highlighting method, or some combination of them contributed to the success of IH and SDN, nor do they show whether the AccessRank parameters were optimal. Although simulation results in Chapter 6 suggest that the use of AccessRank was appropriate, further work is necessary to determine whether this is indeed the case. Chapter 8 includes subjective results on the perceived predictive accuracy of AccessRank predictions for Icon Highlights, however comparative results between different predictive algorithms showing their relative effects on retrieval times are beyond the scope of this thesis.

Limitations of the Experimental Method

Non-natural setting: The experiment was designed to emulate a familiar dataset. This preliminary evaluation answered specific research questions that confirmed the techniques' feasibility. However, a large field study involving participants' own personal file collections would yield more accurate findings; such a study is described in Chapter 8.

Task cuing strategy: The evaluation method initiated tasks by showing a stimulus consisting of the target file's name, which allowed users to simply

type the name for a certain match with SDN. In reality, recall of filenames is imperfect (Blanc-Brude and Scapin study [32] found that correct filename portions – necessary for use of SDN – were recalled for 83% of files, though only 25% of filenames were perfectly recalled). While use of filenames as task stimuli is not a perfect solution, it was the most appropriate given the constraint of an unfamiliar document collection; the difficulties in setting appropriate cues for file retrieval tasks in lab studies are well documented [104, 27]. Because of this limitation, however, the results for initial selections with SDN can be considered a best case analysis. This limitation is also resolved in the field study in Chapter 8.

SDN vs search: The evaluation compared performance between alternative interfaces supporting navigation-based file retrieval. Performance comparisons between search and navigation were not addressed. A future evaluation could directly compare how search and SDN influence file location learning, whether they have different perceived workloads, and which techniques are best in different circumstances, however this is beyond the scope of this thesis.

Sample size: While the results were highly significant, a larger sample size would produce more robust findings. There is little reason to doubt the key results based on this limitation, although a larger sample may have affected some of the more subtle differences between techniques in specific situations.

Erroneous predictions: While the experiment design did not allow for direct analysis of the effect of erroneous predictions, their effect was incorporated into the main results, showing that the interfaces still provided significant overall benefits. Further studies are needed to investigate their effect in more detail.

Variation in users: Further work is required to investigate the difference in effectiveness of the interfaces for users who organise their hierarchies in different way (e.g., pilers vs filers [127]). Previous results have suggested differences in both organisation and retrieval behaviour between filers and pilers [94, 183]. These differences affect properties such as folder breadth and retrieval frequency, which could affect the quality and usefulness of predictions.

7.6 Conclusion

This chapter introduced three interface designs to improve user performance in accessing files by navigating through hierarchical structures. The designs use underlying algorithms or search queries to predict likely target files, and they use these predictions to assist users in traversing the hierarchy. The interfaces are designed to *assist* with hierarchical traversal, rather than simply deliver a list of likely targets, because they are intended to help users learn the location of files, and the associated mechanisms for retrieval, to assist with future navigation-based file accesses.

The interfaces are derived from three design goals: (1) minimise the time spent at each hierarchical level; (2) reduce the number of hierarchical levels that must be traversed; and (3) promote rehearsal of the retrieval mechanics to facilitate expertise. The Icon Highlights interface is designed to assist users in visually identifying likely targets at each level of the hierarchy (goal 1). Evaluation results show that it performs particularly well when users are revisiting files in folder views that are spatially unstable. Hover Menus are designed to facilitate shortcuts to likely folders and files across levels of the hierarchy (goal 2), and results showed that they are particularly effective for revisiting files in spatially stable views. Search Directed Navigation highlights items that match search query terms, emphasising those that have been previously visited. It offers an alternative to search by guiding users through the hierarchy to matching items, forcing rehearsal of the navigation actions used when expert (goal 3). Results showed that it performs particularly well when users do not have location knowledge.

All of the interfaces allowed faster task completion than the standard file browser in all conditions, and subjective preferences favoured them. However, this chapter only evaluated them in a lab setting, which involved a variety of methodological constraints. To confirm their benefits, Chapter 8 further develops two of the techniques and evaluates them in an externally valid, natural setting. Nevertheless, these initial results suggest that these relatively easy to implement features can greatly improve the common activity of navigation-based file retrieval.

Chapter VIII

Finder Highlights: Design and Evaluation of an Augmented File Browser

Chapter 7 described the design and evaluation of three techniques for use in a standard file navigation interface: Icon Highlights, which highlight items in the current folder that are most likely to be accessed next; Hover Menus, which show a menu providing shortcuts to highly ranked descendant files and folders when hovering over a folder icon; and Search Directed Navigation, which highlights items that match, or contain items that match, a search query. The evaluation showed that each is effective at reducing retrieval times, with Icon Highlights and Hover Menus specialising at revisitations, and Search Directed Navigation specialising at locating unfamiliar items.

These findings are important in suggesting the potential for improving a core interaction that is used many times a day by most computer users. However, the results only demonstrate a *potential* – the techniques improved file retrieval performance when used in a lab setting that applied a wide set of experimental constraints on their use. Factors that limit the external validity of the findings, or that raise risks of generalisation, include both technical aspects of system behaviour and methodological constraints.

The technical aspects are related to the design of the browser used for the evaluations; it was intentionally rudimentary in order to limit interaction to only the key interface features under study. In particular, the following differences existed between the evaluated browser and real-world file browsers:

Tools – the evaluated browser eliminated navigation tools normally available in fully functional file browsers, such as shortcuts, search features, and navigation buttons.

Views – the evaluated browser provided a simple icon view, displaying items in grid formation, whereas real-world file browsers normally allow extensive view configuration.

Integration of techniques – a real-world file browser might be expected to support the techniques simultaneously, but the evaluated browser supported only a single technique at any given time.

Naïve back-end algorithms – the evaluated systems made unrealistic simplifying assumptions to enable prediction of upcoming file selections, and did not have to consider performance issues due to the artificially small hierarchy containing only 836 nodes (files and folders).

On the methodological side, as with any lab study, the empirical methodology constrained external validity.

Artificial and static file hierarchy – the artificial file hierarchy of static content differed from the user's real file system in terms of size, structure, content, and importantly, familiarity.

Specific tasks – tasks were limited to file retrievals, ignoring other tasks performed with file browsers such as file organisation.

Revisitation patterns – participants were required to repeatedly revisit files within a short experiment, creating unrealistically dense revisitations.

Task stimuli – rather than retrieving files when needed for actual work, participants retrieved them in response to a filename cue, when other cues may be more realistic [32].

Discrete tasks – each retrieval began at the root level of the file hierarchy, ignoring the possibility of window reuse.

The evaluation in Chapter 7 provided important results about the potential for Icon Highlights, Hover Menus and Search Directed Navigation.

However, the above issues must be addressed before the potential of these interfaces can be realised in real-world file browsers. This chapter therefore describes the design, implementation and longitudinal evaluation of a plugin to OS X's Finder, the operating system's standard file manager. The plugin, called Finder Highlights, introduces support for Icon Highlights and Search Directed Navigation – the two techniques showing the most promise.

Particular focus is given to the algorithmic challenges of these implementations: for Search Directed Navigation, efficiently determining which icons to highlight, and adapting the results to query or location changes without restarting the entire search; for Icon Highlights, modifications to AccessRank (Chapter 6) to better support the domain of file navigation, including use with hierarchical data. The chapter also describes a potential implementation of Hover Menus, including a description of *Benefit Weighted AccessRank*, a modified version of AccessRank that tailors results based not just on which items are most likely to be accessed, but also the degree of benefit provided by potential shortcuts to each item.

The chapter then describes a four-week longitudinal field study of Finder Highlights. Results confirm that both Icon Highlights and Search Directed Navigation can improve file retrieval times and are subjectively preferred over standard techniques. Analysis of file retrieval performance shows that retrieval times when using Finder Highlights (mean 10.6 s) were 13% faster than retrievals accomplished without it (mean 12.2 s). However, many participants did not use SDN, associating it with rarely-used search interfaces. This distinction between favourable lab study results for SDN (where all participants employed its features and benefited from them) and the field study results (where many participants ignored it) validates the motivation for the study. The finding also corroborates several prior studies showing that lab studies often produce more positive results than field studies [43, 60, 96, 146].

The chapter concludes by discussing the implications of the results and giving direction for future refinements.

8.1 Finder Highlights

Finder Highlights is a fully functional plugin to the standard OS X file browser, the Finder, that adds support for Icon Highlights and Search Directed Navigation. The design and implementation of Finder Highlights serves two purposes. First, to confirm that the techniques are feasible outside a prototype system, both in terms of integrating the techniques into an existing file browser user interface, and developing underlying algorithms that efficiently determine highlights. Second, to investigate whether the potential benefits of the techniques, shown by the results of the lab study in Chapter 7, are realised in real-world file retrieval tasks.

Finder Highlights includes support for two out of the three techniques described in Chapter 7: Icon Highlights and Search Directed Navigation. Focusing on these two techniques allowed for more in-depth analysis on how they are used; including Hover Menus would make it difficult to tease apart the relative benefits of the techniques, as there would be complicated interactions between them that would be difficult to analyse in a field study. There were several reasons for the selection of Icon Highlights and Search Directed Navigation as the two techniques to implement. First, these were the two fastest methods in the lab study, particularly when item locations were not spatially stable. Second, they both use a highlighting-based approach, introducing interesting design questions about how they can be implemented together. Third, Icon Highlights serves as a *passive* technique, with its highlights appearing automatically with no user input required, whereas Search Directed Navigation is an *active* technique that requires explicit action before it can be used. This contrast makes for interesting comparisons between a lab study, where participants may feel obligated to use new or novel interaction techniques, and a field study, where active techniques may be less likely to be used [43, 96]. Fourth, Icon Highlights and Search Directed Navigation were designed for different types of retrieval tasks (Icon Highlights for revisitations, Search Directed Navigation for hard-to-find items), so there is likely less interaction between the techniques as there would be between Hover Menus and the highlighting-based interfaces.

A high-level architecture of Finder Highlights is shown in Figure 8.1. It

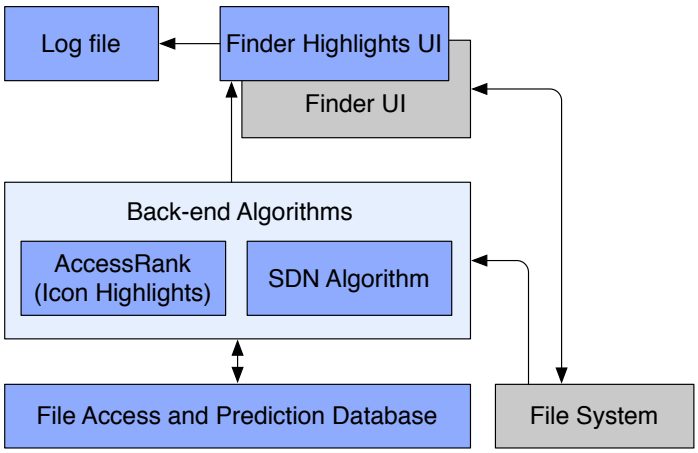


Figure 8.1: High-level architecture of Finder Highlights. Grey cells represent system components, blue cells represent parts of Finder Highlights, and arrows indicate information flow.

consists of a user interface layer (Finder Highlights UI) that extends the OS X Finder. The user interface layer requires file access predictions, which are provided by two different algorithmic components. The first of these serves Icon Highlights predictions, and uses modifications and extensions to AccessRank that were required to enable AccessRank to support mutable, hierarchical data (e.g., to support score changes when a file is moved between folders). The second component determines search results for Search Directed Navigation. Both the Finder and the predictive algorithms draw on data from the file system. The algorithms store data about retrievals and predictions in a database, and all user interactions with the file browser are written to log files.

The design and implementation of Finder Highlights are described in the sections below, split by technique: first Icon Highlights, then Search Directed Navigation.

8.2 Design and Implementation of Icon Highlights

The implementation of Icon Highlights as part of Finder Highlights involved two main areas of work: *user interface* refinements for use in a real file browser, and *algorithmic* enhancements to AccessRank to support and enhance use with a file hierarchy. These are detailed below.

8.2.1 Interface Design

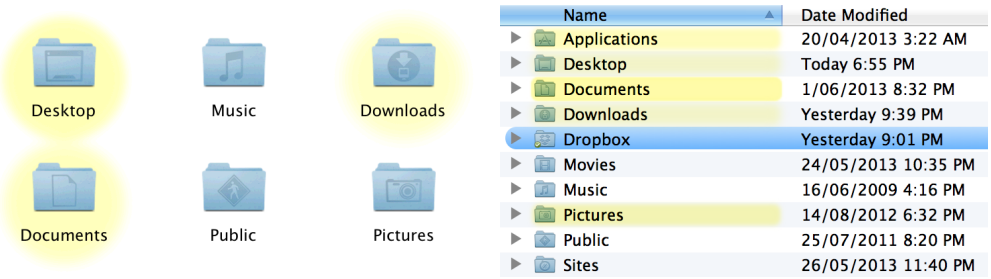
In Chapter 7, both Icon Highlights and Search Directed Navigation were implemented as circular highlights cut out of a transparent grey overlay. The underlying file hierarchy was always presented as a grid of icons, much like the icon views in OS X and Windows. While this design was sufficient in the context of the initial lab study, a real-world implementation presents several design challenges:

- Challenges in *supporting multiple views*; the original circular highlight would not work for views that display item in a list, for example.

- Challenges in *supporting Icon Highlights and Search Directed Navigation simultaneously*, ensuring that the highlights presented by the one technique are not confused with those of the other.

- Challenges in *competing with other visual features*, ensuring that highlights do not diminish or obscure other features such as colour labels or icons.

These challenges were addressed as follows. Highlight shapes, sizes and blur levels were adjusted to support multiple views. Highlights displayed using Icon Highlights were modified to use a yellow spotlight rather than a white cutout. This distinguished the highlights from those used by Search Directed Navigation, which maintained the original cutout method. Additionally, it eliminated the permanent grey overlay for Icon Highlights, where a permanent loss in contrast would be disadvantageous (on the other hand, its temporary use to highlight items in response to explicit user actions, as in Search Directed Navigation, is appropriate). To ensure that the yellow highlights do not interfere with other visual features such as colour labels, the highlights are drawn behind items where there is sufficient space (i.e., in icon view), and above them with a blend function that ensures visibility of underlying content in all other cases. These refinements are detailed below.



(a) Icon view (b) List view

Figure 8.2: Icon Highlights shown in different views.

Highlight Appearance for Icon Highlights

Icon Highlights uses yellow highlights to identify likely items. An item's AccessRank score determines both the blur level of the highlight (in the same way as in Chapter 7) and the brightness of the highlight colour. RGB values for the highlight colour are calculated as $(1,\ 1,\ 0.55 + 0.2(\frac{b}{b_{max_i}}))$, recalling that b is the highlight's blur level and b_{max_i} is the highlight's maximum blur level, and where (1, 1, 1) corresponds to white.

Support for Different Views

The preliminary systems described in Chapter 7 only supported a standard icon view. Finder Highlights supports all of the Finder's views, regardless of the icon and text sizes used within them. Figure 8.2 shows two forms that highlights can take with Icon Highlights, while Figure 8.3 (later in this chapter) shows a wider range of appearances with Search Directed Navigation (the size and shape of the highlights is the same for both techniques).

For icon view, highlights have a similar circular design to those in Chapter 7. The exception is when filenames are shown on the right of icons, such that the total area containing the icon and filename is much wider than it is high. In these cases, the highlights are displayed as rounded rectangles. Highlights can also overlap each other if icons are close together (seen, for example, with the 'Desktop' and 'Documents' highlights in Figure 8.2a).

For column view, a shorter, wider rounded rectangle highlight appears around the item's icon and filename label. This is similar for list view (Figure

8.2b), including Cover Flow view, noting that content outside the 'Name' column is not highlighted.

To ensure that these short highlight rectangles have similar visual saliency to larger highlights, the maximum blur level for highlight region i is reduced for small highlight regions as follows, where d_{min} is the smallest dimension of the highlight region, in pixels, and b_{max} is the global maximum blur level:

$$b_{max_i} = \min(b_{max}, \frac{d_{min}}{48} b_{max}) \tag{8.1}$$

Highlight Rendering

As well as adjustments for size and shape, highlights are rendered differently in icon view (Figure 8.2a) and other views such as list view (Figure 8.2b). In icon view, there is a large amount of whitespace surrounding each icon. If an item has been given a colour label, that label appears behind only the icon's filename, preserving most of the whitespace. The highlight can therefore be drawn behind the icon and label, while still having guaranteed visibility. Other views, however, have less whitespace, and any colour labels generally consume all of it (for example, 'Dropbox' in Figure 8.2b). In these views, highlights are drawn *on top* of the underlying view, using a blend mode that ensures the highlight is visible without affecting the readability of the text or other content underneath. In this blend mode, which uses a subtractive model, each component of the resultant colour is calculated as $\max(0, 1 - (1 - c_1) - (1 - c_2))$, where c_1 and c_2 are the corresponding components of the two colours being blended.

Highlight Selection

A maximum of seven highlights are shown for each location, corresponding to the top seven AccessRank predictions – though, due to its stability component, these are not necessarily the items with the highest scores. This maximum ensures that Icon Highlights' benefits are not diminished over time as a wide range of files are accessed, which would otherwise result in a majority of items being highlighted in some locations. Note that in some cases multiple locations are visible in a single view at once, such as when a folder is

expanded in list view or multiple columns are visible in column view. In these cases, predictions are made separately for each location, and the maximum number of highlights is applied to each, rather than acting as a global maximum. Icon Highlights are only displayed when Search Directed Navigation is not active.

8.2.2 Extending AccessRank for use with Icon Highlights

Icon Highlights uses a modified version of AccessRank (Chapter 6) to determine which items to highlight, and how prominently to highlight each one. These modifications are necessary to support predictions in a changing hierarchical structure where only a subset of files and folders are displayed at a time. Further changes are also made to reduce the influence of file retrievals made outside of the Finder; as found in Chapter 6, an item's access history using one method has little predictive power on whether it will be accessed with another method.

Configuring AccessRank parameters for Icon Highlights

AccessRank has two parameters, λ and δ, that should be calibrated for each domain it is used in. Comparisons of accuracy and stability results for AccessRank configurations on the file navigation dataset from Chapter 5 suggest that AccessRank 2 with (λ, δ) = (1.65, 1) is an appropriate configuration, with low average rank, a high percentage of revisitations predicted, and high Rank-Biased Overlap (RBO) scores, collectively ensuring both high accuracy and high stability. The λ value of 1.65 indicates that the algorithm's Markov component, which is heavily dependent on the most recent retrieval, has a higher weight than its Combined Recency and Frequency (CRF) component, which incorporates both recency and frequency. This is beneficial, since Chapter 6 found that recency is a much stronger predictor than frequency for the domain of file retrieval. For practical reasons, AccessRank's time component was not incorporated, as this would have had a large effect on the amount of history data that would need to be written to disk, and prior results indicate that its inclusion produces relatively small benefits.

Adaptations to support location-specific predictions

AccessRank ranks only the items in the current location when making predictions for Icon Highlights. These predictions can be made for either a fixed location (e.g., a folder) or a dynamic location (e.g., the results of a search).

AccessRank scores can be efficiently updated to enable quick location-specific predictions. The model uses a global event counter that is incremented for every retrieval. Although item scores generally decrease each time the counter is incremented (due to AccessRank's CRF component), these scores do not need to be updated after every retrieval. Instead, an item's score is updated in $O(1)$ time only immediately before it is ranked; this ranking only occurs when the item's parent folder is viewed.

AccessRank's stability component requires a threshold before two items can change the relative order of their rankings, based on the magnitude of the difference between their scores. Finder Highlights applies this stability component only when viewing a fixed location (i.e., a folder), not when viewing dynamic content such as search results. This enables consistent behaviour, with AccessRank providing extra stability for successive predictions of items within the same parent folder, but not applying the stability component when it makes little intuitive sense (as there is no conceptual link between the content of the results of successive searches). To achieve this behaviour, Finder Highlights updates the previous rank of an item only when it is ranked in the context of its parent folder.

Recording accesses at retrieval-time

Accesses are recorded for file retrievals, with accesses for files' ancestors being recorded simultaneously. This ensures that navigational errors do not affect predictions, as would occur if accesses were recorded for each navigation step (and noted by one participant in Chapter 7).

For AccessRank's Markov model to function, it must have an appropriately set *previous item*. This is set to the previously retrieved file, rather than any folder accessed in an intermediate navigation step. The model is updated for all path components of each retrieval. As an example, if */Applications/Utilities/Terminal* is opened, immediately followed by */Applications/Microsoft*

Office 2011/Microsoft Word, then the Markov model would be updated for the sequences *Terminal* → *Applications*, *Terminal* → *Microsoft Office 2011* and *Terminal* → *Microsoft Word*. The predictions for each step of a retrieval are therefore all based on the particular file that was most recently accessed. In the Markov model, the count for each of these transitions is incremented by one, however the *total* count is only increased by one for each retrieval.

Incorporating retrievals made using other methods

Finder Highlights greatly reduces the influence of retrievals made outside the Finder on predictions for Icon Highlights. This reduction is necessary because results in Chapter 6 indicated that the access history of an item with one retrieval method, such as search, has little predictive power over future accesses with another method, such as navigation. The influence of these retrievals is *reduced*, rather than *removed*, because it would not conform with user expectations if items previously accessed frequently with methods other than Finder navigation were never highlighted with Icon Highlights.

The reduction in the influence of these retrievals is done by weighting them as if they occurred 400 events further in the past than they actually occurred. This results in negligible scores in comparison with items that have been accessed from within the Finder, meaning that items that have only been retrieved in this way have either (1) very dim highlights, when viewed alongside items that have been retrieved through the Finder; or (2) a highlight in the normal range when viewed alongside items which have also not been retrieved through the Finder in some time. In the latter case, relative item prominence is preserved.

Retrievals made outside the Finder are ignored by AccessRank's Markov model. For example, if "Budget.xls" is opened in the Finder, followed by "Todo.txt" using an Open Recent menu and "Bank details.doc" in the Finder, the Markov model is updated as if "Budget.xls" was followed by "Bank details.doc", with no consideration of "Todo.txt".

Within Finder, a subcategory of navigation retrievals are those that use OS X's *Quick Look* feature. Quick Look allows an item to be quickly previewed in a lightweight window within the Finder without opening the item

in a separate application. Once a preview window is shown, users can switch between files using the keyboard, allowing them to quickly preview a large number of items. As a result, use cases for Quick Look range from skimming through a large number of files while searching for a specific target, to using it as a substitute for opening a file. These two extremes have significantly different intents, and it is important that they are handled separately; for example, if Quick Look is used to quickly scroll through an entire folder of photos, it would make little sense to record retrievals for every one, while if it is used to watch an entire video as a substitute to opening it in a media player, it would be an error not to record it as a retrieval. Finder Highlight attempts to distinguish between these cases by recording a retrieval only if the Quick Look preview for a single item is visible for at least five seconds.

Supporting aliases

Aliases and symbolic links provide shortcuts to files and folders located elsewhere in the hierarchy. When accessing an alias, Finder Highlights records a retrieval for the alias's original file (i.e., the file it references), rather than the alias itself. When ranking items, AccessRank uses the score of an alias's original file, although the alias records its own previous rank for the purposes of AccessRank's stability component. These changes ensure that items are highlighted in a way that assists subsequent retrievals of an item, regardless of the path taken to it.

Adjusting scores in response to file management tasks

When a file is retrieved, Finder Highlights records accesses for both the file and each of its ancestors. This means that a folder's AccessRank score is reflective of the scores of all the items inside it. To preserve this property, Finder Highlights updates folder scores when items are moved between folders or deleted.

When an item is moved or renamed, it carries its own score and history data to its new path, and references contained in the Markov model are updated. Markov transitions are stored in memory as two way links for efficient updates, so that each item is aware of both which paths have been

retrieved following its retrievals, and which paths it followed. When an item is deleted, its data is removed.

Ancestors of moved or deleted items have adjustments made to their CRF weights. Adding an item to a folder increases its CRF weight, while removing an item from a folder either decreases its CRF weight or leaves it the same, depending on the type of item. Conceptually, items are classified as follows:

Files are leaf nodes in the hierarchy, whose CRF weights correspond only to their direct access history.

Containers are the direct parents of files. If a container's children are moved or deleted, it may still be an area of active work, with activity simply transitioning to other files in the same area.

Intermediaries are the parents of containers or other intermediaries. If an intermediary's children are moved or deleted, this may be more indicative of a change in the hierarchy structure, rather than the particular file makeup of an area of current activity. Note that a folder that contains both files and folders may act as either a container or an intermediary, depending on the context.

Accordingly, containers and intermediaries are treated differently when updating CRF weights. If the child of an intermediary is deleted or moved to another location, Finder Highlights reduces its CRF weight. If the child of a container is deleted or moved, its CRF weight is *not* affected, since it may still be an area of current activity.

Adjustments to folders' CRF weights are made by adding or subtracting the CRF weights of the descendant items that are being added or removed. This is due to the summative nature of CRF weights – the CRF weight of a folder is generally equal to the sum of the CRF weights of its children. All changes are performed recursively up the hierarchy. Before any CRF weights are modified, weights of all applicable items are updated based on the current CRF event counter.

Accounting for simultaneous accesses

The Finder allows users to select and open multiple files simultaneously. An example of when a user might choose to do this might be to open a set of images in an application that allows them to easily view or compare them. Ordinarily, these files would be treated as separate retrievals, and the files' ancestors would receive excessively large scores relative to their siblings, even though the change is conceptually the result of a single retrieval task. Instead, when multiple files are opened simultaneously, Finder Highlights records only a single access for each of their common ancestors, while still recording a retrieval for each of the individual files. This is the only situation in which the CRF weight of a folder can be less than the sum of the CRF weights of its children. Importantly, this behaviour is allowable since moving or deleting a file, as opposed to a folder, does not affect the score of its original parent, and because simultaneous accesses can only occur for a set of files; were this not the case, a folder could conceivably have a negative CRF weight after removing descendants that were opened simultaneously.

8.3 Design and Implementation of Search Directed Navigation

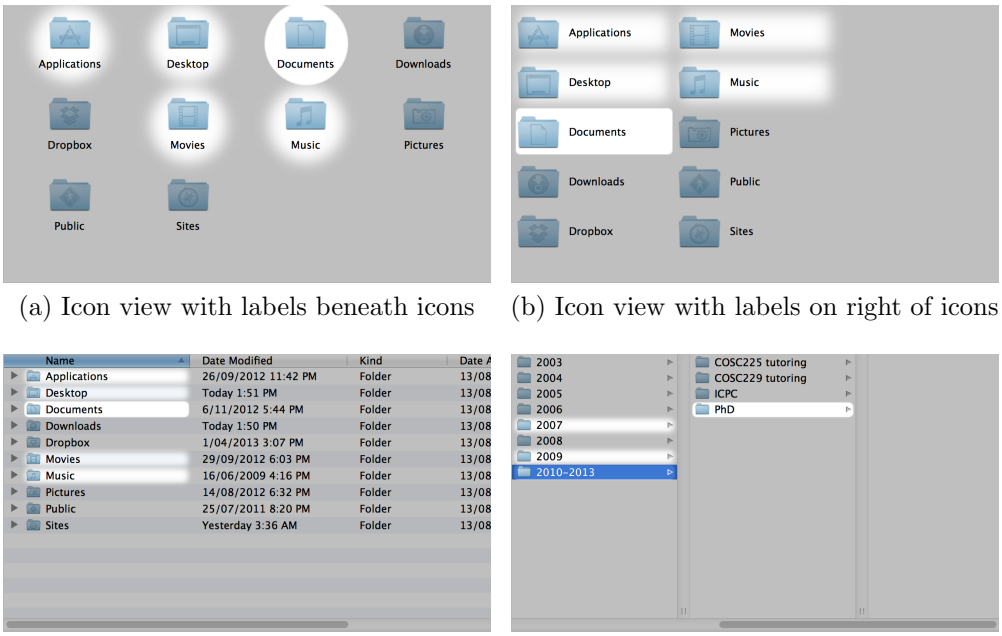
As with Icon Highlights, Search Directed Navigation required both user interface refinements and algorithmic enhancements. These are described below.

8.3.1 Interface Design

The interface design for Search Directed Navigation shared many common challenges with those of Icon Highlights: supporting multiple views, supporting use of both techniques simultaneously, and competition with other visual features. Additionally, it introduced an additional challenge: integrating the Search Directed Navigation search field with that of the existing search functionality in Finder.

Highlight Appearance and Implementation

As with Icon Highlights, Search Directed Navigation alters highlight appearance based on the folder view, but highlights are otherwise styled in the same



(a) Icon view with labels beneath icons

(b) Icon view with labels on right of icons

(c) List view

(d) Column view

Figure 8.3: Search Directed Navigation highlights shown in different views

way as in Chapter 7 (i.e., as a white highlight cut out of a transparent grey overlay). Highlights are shown with either crisp borders (previously visited items) or blurred borders (unvisited items), as in the original design. Figure 8.3 shows some of the forms these highlights can take.

The Search Directed Navigation view is implemented as an overlay window which updates whenever the underlying content updates, whether it be a content update (navigating to another folder, or a change in the content of the current folder) or a view update (scrolling, resizing, changing view types, or changing view properties such as icon size, text size, or grid spacing). Implementing it as an overlay makes it easy to alter the view's appearance while maintaining a strong degree of separation from the underlying view, which is beneficial for plugin stability. The overlay itself is slightly darker than that used for Icon Highlights and Search Directed Navigation in Chapter 7, to improve the contrast between highlighted items and other content.

In large views, highlights may appear outside the visible region. While possible approaches to indicate the existence of these highlights are discussed

later in the chapter, Finder Highlights includes one refinement to reduce the effect of this issue. When opening a folder while Search Directed Navigation is active, the view will automatically scroll so that the first result is visible; if one is already visible, this will have no effect. If a match is found only after the folder is opened, no scrolling occurs, as this may be disorienting to the user.

Search Field

Search Directed Navigation and Finder's search feature offer differing use cases. For example, Search Directed Navigation is specialised for filename queries, whereas search supports a range of file attributes. Both techniques are activated by typing a query into a search field. Providing a separate search field for each of the two techniques would clutter the browser interface and would be likely to confuse users, so Finder Highlights integrates Search Directed Navigation within Finder's existing search field in its toolbar (Figure 8.4).

The search field is live updating, meaning that it begins searching immediately when text is entered, rather than waiting for a button click or for the enter key to be pressed. Finder Highlights adds a menu to the search field, activated by clicking the search icon at the left of the field, which can be used to switch between the existing Finder search and Search Directed Navigation. When no query has been entered, the search field displays the current search mode as dimmed placeholder text (Figure 8.5), showing either "Search Directed Navigation" or "Finder Search". Switching between search modes cancels any current search and clears the search text.

Search Directed Navigation's grey overlay (and any search results) are displayed only when text is entered in the search field while Search Directed Navigation is the active search mode; it is not displayed when the search field is empty. Yellow highlights from Icon Highlights are displayed only when the Search Directed Navigation overlay is not shown.

New Finder windows initially show Search Directed Navigation as the active search mode in most cases. The exception is windows created in response to a search related event (for example, by choosing "Show All in Finder" from

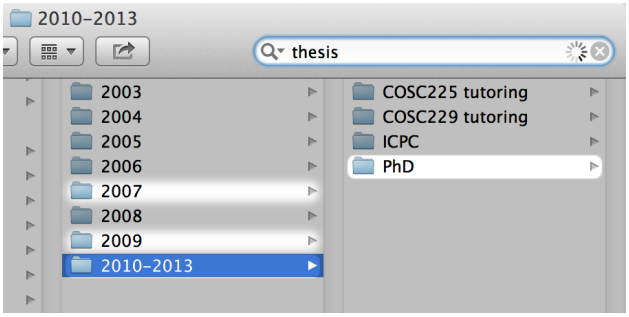


Figure 8.4: The Finder toolbar's search field, used to enter the SDN query. Clicking on the search icon at the left of the field shows a menu which toggles between search modes.

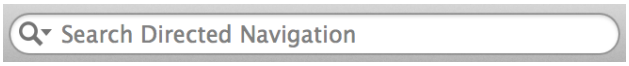


Figure 8.5: The search field, using the placeholder text to indicate the current search mode.

the Spotlight menu or typing a keyboard shortcut for search), which instead show Finder Search, for consistency with previous behaviour. Once a window is active, command-F (the keyboard shortcut for *Find*) activates the search field without changing search modes. This ensures that the placeholder text users see in the search field is consistent with the search mode that is activated when they move focus to the search field via the keyboard. Another shortcut – command-option-F – is provided to toggle between search modes and move focus to the search field (if not already focused). Using it when no windows are open has the same effect as using command-F.

Figure 8.4 shows a Search Directed Navigation search that is currently in progress. The cancel button (far right of the search field) immediately ends the search, clears the search field, and hides the Search Directed Navigation overlay. The progress indicator (immediate left of the cancel button) is shown while there is still the possibility of changes to the set of highlighted items in the current location. Searching may continue after it has disappeared, however this searching will be at deeper locations in the hierarchy where any visible ancestors have already been highlighted. Additionally, the progress indicator may disappear before highlights for certain low interest locations

have been finalised, such as system folders. The effect of this behaviour is that the progress indicator acts as a way for the user to know whether there is a chance for matches of other visible locations if they continue to wait, or if the current set of highlights constitutes all matched subtrees of the hierarchy.

8.3.2 Search Directed Navigation Algorithm

Search Directed Navigation provides visual feedback that guides users to descendent files and folders that match a search query at any lower level of the hierarchy. As a result, its underlying algorithm must efficiently cope with a potentially large search space. While the algorithm in the lab prototype (Chapter 7) was able to perform an exhaustive search in real time, actual file hierarchies are much larger and require a more sophisticated approach. This section therefore describes a specialised algorithm that is used to determine matches for Search Directed Navigation.

A variety of goals were considered in the design of a suitable algorithm:

- Quick access to previous items: most retrievals are for previously visited items, but these make up only a small proportion of all files. These files should therefore be searched and displayed first.

- Fast feedback: the priority is to quickly determine which items in the current view contain matched descendants, rather than to determine *all* matched descendants of visible items. However, all matched descendants should be recorded, when found, to facilitate responsiveness when subsequently navigating down the hierarchy. User feedback should be provided once the set of highlights for currently viewed items is finalised, even if all their descendants have not yet been checked.

- Platform independence: the algorithm design should not be tied to a particular platform or system.

- Tolerance of imperfect information: it would not be possible to create and maintain an index of the entire file hierarchy, as the plugin does not have access to low level system information about every change to the file system, and thus the index would become outdated. While it

has access to some changes made within the Finder itself, changes made from other applications are not recorded. Additionally, new devices can be connected, and the algorithm should be able to search their content without requiring time to first index them.

- Interruptible: the search should be interruptible, so as not to interfere with user input when searching a large hierarchy subtree.

- Adaptable to changing location: the search should adapt to changes in the window's hierarchy location without restarting the entire search, where possible.

- Adaptable to changing query: the search should adapt to changes in the search query without restarting the entire search, where possible.

- Adaptable to changing content: the search should adapt to changes in the folder hierarchy it is searching, ensuring that any new content is searched, and any previous matches are updated in response to moved or renamed content.

In order to satisfy these requirements, the implementation combines several methods to achieve the best results, as visualised in Figure 8.6. The core algorithm is composed of two phases. In its first phase (the *revisitation index*), it searches an index of previously visited files and folders, returning matches for these items almost instantaneously. In its second phase (the *exhaustive search*), it performs a breadth-first search of all descendants of the visible items.

The time it takes for the exhaustive search to complete varies considerably based on the context of the search. In the worst case – when searching an entire system – this can take about half a minute to complete on a typical modern computer. However, in practice, most searches will match multiple files, and the first results are typically shown much sooner – the effect being that many searches have a complete set of highlights determined in less than a second. Nevertheless, the exhaustive search method does result in an undesirable worst case performance, exhibited when searching the entire system

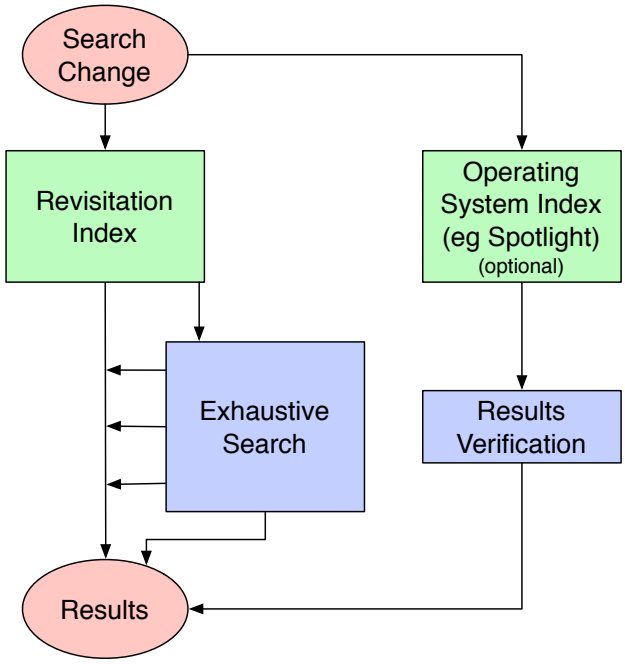


Figure 8.6: An overview of how Search Directed Navigation results are determined, using a combination of different methods.

for a unique filename located deep in the hierarchy in a low priority location that has not previously been visited.

To address this issue, the system's search index (if available) can be used to find results simultaneously to the core algorithm. OS X's Spotlight index provides an example of such an index, and it was used in Finder Highlights. The system search index serves as an optional addition to improve performance, but is not required. The design supports use of such an index to produce a set of potential results, which are then verified based on the requirements of the core algorithm; the system index therefore need neither produce an exhaustive nor an accurate set of results, however it can significantly improve performance by approximating both.

Full details of this process, as well as information about how the algorithm copes with changes to its query, location, or content, are described below.

Matching items

In Chapter 7, an item was defined as a match for Search Directed Navigation when any word (or consecutive combination of words) from the filename had the search query as a prefix. This definition was expanded for the real-world implementation, due to the wider range of ways that tokens are delimited in practice, compared to the dataset used for the prototype. In fact, results from Chapter 5 showed that spaces, hyphens, underscores and periods were all common, as was mixed case.

To determine matches, filenames are first decomposed into a set of match keys, with a match occurring when a match key contains the search query as a prefix (ignoring case). Each key consists of a filename substring, beginning at the start of a token (defined below) and continuing until the end of the filename. This allows multiple tokens to be entered as a query, as in the prototype, provided they are delimited in the same way as in the filename. File extensions are not included as a key themselves, but are included at the end of other keys.

The start of a token is defined as either: (1) the first character of the filename; (2) any non-delimiter character following a delimiter; (3) an uppercase character following a lowercase character; or (4) any change between character types, excluding a delimiter following another character. Delimiter characters include a space (‘ ’), hyphen (‘-’), underscore (‘_’) or period (‘.’). Character types are letters, numbers, delimiters and all other characters. The ' character can act as either an apostrophe or a quotation mark, and is therefore included in both the set of letters and other characters. This means that it can only be the start of a token if it follows a number or delimiter. In such cases, it is treated as a member of the ‘other character’ set, and any subsequent letters begin a new token. With this behaviour, “Today's tasks” would have keys “Today's tasks” and “tasks”, “Photos '09” would have keys “Photos '09”, “'09” and “09”, and “Filename 'quote'” would have keys “Filename 'quote'”, “'quote'”, and “quote'”. As a longer example, “TravelBudget13_Spain trip.xls” would have keys “TravelBudget13_Spain trip.xls”, “Budget13_Spain trip.xls”, “13_Spain trip.xls”, “Spain trip.xls” and “trip.xls”. Any query that is a prefix of one of these keys would result in a match.

Index of previously visited locations

Finder Highlights maintains its own index of previously visited files, allowing for quick lookup of the most likely matches. Like in the prototype implementation of Search Directed Navigation, these matches (i.e., revisitations) are shown with a crisp border. Although some systems feature their own search indexes that can optionally be used in tandem with Finder Highlights' own index, maintaining its own index provides several benefits. First, it decouples its algorithms from a particular platform. Second, it acts as a more lightweight index, containing only those items that have been previously visited, and is optimised for the behaviour of Search Directed Navigation – meaning that the most likely matches can be returned faster than they would when using an external index.

The index contains three types of locations, which are treated identically: (1) files that have been previously opened; (2) folders that have been opened in the Finder; and (3) all ancestors of other locations in the index. Items are added as they are opened in the Finder, or in the case of files opened through a method outside the Finder, when they are added to the user's recent items file (see Section 4.2.4).

The index updates automatically as a result of items being moved, renamed or deleted within the Finder. Some file system changes can occur outside the Finder, and these changes are not detected by Finder Highlights. Instead, every time an item is accessed in the index, a check is performed to ensure it still exists in the file system; if it does not, then it is removed from the index.

The index is stored on disk as an alphabetically sorted list of paths. Once loaded into memory, the index is stored in three separate collections: (1) the path list: a list of path objects, sorted by path, each storing the original path and a set of match keys for each; (2) the index: a list of all key-path pairs, sorted by key; and (3) a mapping from string paths to path objects.

To find matches for a query, a binary search is performed on the index to find the first and last matches; a match occurs when the match key has the query as a prefix. This search does not consider location; matches will be returned regardless of whether they are descendants of the search

location. Matched items are inserted into a tree of results, mirroring the actual file hierarchy, which allows for quick lookup of matches in a particular location. When adding a path, nodes are added for ancestor paths if not already present in the results tree. This entire index lookup operation takes $O(\log(n) + m)$ time, where n is the number of key-path pairs in the index, and m is the number of matches for the query.

Exhaustive search of other items

While the index of previously visited items quickly finds the most likely results, an exhaustive search is still necessary to find other matches. This is done using a breadth-first search of the visible subtrees of the file hierarchy, with the search beginning following the completion of the index lookup. The search gives priority to locations that do not yet have any results, as well as to certain pre-defined common locations.

A breadth-first search is used to reduce the effect of particularly large subtrees slowing down discovery in other locations, as might occur with depth-first search. This reflects the primary goal of the exhaustive search: to quickly determine which of the visible items should be highlighted, rather than minimising the total search time. As the user will likely be traversing through the hierarchy as they search, this provides the greatest appearance of responsiveness throughout the process. Further aiding this goal, when a match is found inside a visible folder, any additional items within the same sub-hierarchy are added to the search queue with a low priority. Items are dequeued in strict order of priority, with the time an item is enqueued considered only for items of the same priority. The effect of this is that the algorithm prioritises finding the *first* match for each visible folder. An emergent property of this behaviour is that searches appear to be significantly quicker when there are many results for a query.

The initial items in the search, referred to as *root nodes*, are each of the items in the current location – for example, the ten folders visible in Figure 8.3a. Other items in the current view that are not children of the current location, such as those in previous columns in column view, are not searched, however any existing results inside them are preserved, and any changes to

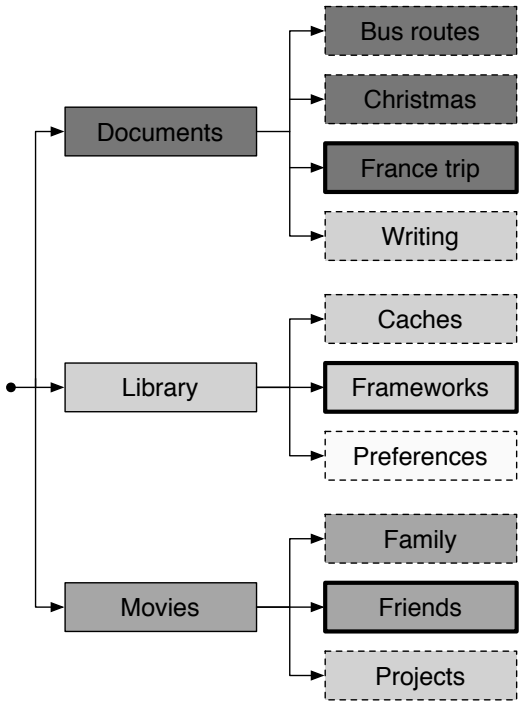


Figure 8.7: Example of exhaustive SDN search for query 'fr' with items of different priorities. Darker shades denote higher priorities. Thick borders denote matches, thin borders denote ancestors of matches, and dashed borders denote all other items.

their highlights (for example, if they are ancestors of the selected folder) are reflected visually. An an example, in Figure 8.3d, the root nodes are the four items inside the selected '2010-2013' folder. The '2007' and '2009' folders are not searched as they are not children of the current location, but because they were part of the existing result set before the '2010-2013' folder was selected (see *Dynamically changing the search location* below), their highlights are preserved.

When a match is found, it is added to the existing results tree that was created during the revisitation index lookup. Nodes in the tree keep track of their score, corresponding to whether they should receive a prominent or dim highlight; this score is the highest of their own score (if the node is itself a match for the query) and those of all their children.

Certain locations are given higher or lower initial search priorities. High

priority locations are the user's documents folder, their downloads folder, and their applications folder(s). However, after finding initial matches in these locations, additional items revert to the same low priority as those in other locations with matches. Low priority locations are the 'System' and 'Library' folders, which contain system files and application data. If matches are found inside these locations, additional items have the lowest priority of all items.

An example search is illustrated in Figure 8.7. The three root folders denoted in the example all contain a child that matches the search query, however they each have different priorities, as outlined above. In this example, the nodes would be evaluated in the following order (ignoring descendant items not shown in the diagram): 'Documents', 'Bus routes', 'Christmas', 'France trip', 'Movies', 'Family', 'Friends', 'Library', 'Writing', 'Projects', 'Caches', 'Frameworks' and 'Preferences'.

Integration with system indexes

Search Directed Navigation's algorithm can optionally use a system-specific index to improve search performance. The Finder Highlights implementation uses OS X's Spotlight index in tandem with the core Search Directed Navigation algorithm in order to find matches more quickly. Following completion of an index search, all results are verified to ensure that they conform to the requirements of a Search Directed Navigation match, and are then added to the results tree. Consistent with the goal of producing relevant results as soon as possible, the index is searched for all filenames inside an observed location that begin with the search query; while this omits filenames that contain the query elsewhere in the filename, it is considerably quicker to check for prefixes, and additional results will be detected with the core algorithm. It is also paramount to minimise the time spent searching the Spotlight index, so as not to noticeably slow down the core algorithm.

In addition to non-prefix matches, some locations may be excluded from the Spotlight index. The core algorithm, while slower to complete, acts as a fallback in all these cases, and indeed, it can work completely independently of Spotlight.

Dynamically changing the search location

Search Directed Navigation is designed for users to navigate through the file hierarchy as they use it. As such, it is important that it can efficiently update its results in response to these location changes, without restarting the entire search. As the Finder Highlights index returns all results across the entire file hierarchy, location changes primarily affect the exhaustive search. Any system-specific index lookup may also need to be repeated or refined with the updated location, although this is outside the scope of the core algorithm.

A location change can be divided into one of three categories: (1) moving up the hierarchy, such as to a parent folder; (2) moving down the hierarchy, such as into a child folder; or (3) moving to an unconnected part of the file hierarchy. Each case is handled differently.

In the first case – moving up the hierarchy – all existing results are maintained, however the search now covers a larger area. The items in the new location are added to the set of root nodes and to the search queue of the exhaustive search, restarting the queue processing if it has previously completed. To ensure that the previous root nodes are not traversed a second time, they remain in the set of root nodes, and only non-root nodes are added to the search queue in later processing.

The second case – moving down the hierarchy – involves modifying the existing search queue, which may now include items outside the scope of the new location. Queue processing, if running, is paused while this occurs. Two cases can occur when modifying the queue. If an ancestor of a new root node exists in the queue, then the new root node has not yet been explored – this might occur when traversing several levels at once, for example by following an alias. In this case, the new root node is added to the queue. Alternatively, no ancestor of a particular new root node exists in the queue, implying that it is currently being searched or has already been searched. In this case, the new root node in not added to the queue, to prevent a second traversal. In either case, nodes that are not descendants of a new root node, or new root nodes themselves, are removed from the queue.

The final case – moving to an unconnected part of the file hierarchy – requires the search to be restarted with the new location.

Dynamically changing the search query

Consistent with Raskin's recommendation for incremental search [154], Search Directed Navigation updates its results instantly whenever its search query is modified. This enables users to adapt or refine their search query based on preliminary results.

Like location changes, query changes can be divided into three categories: (1) appending text to the query; (2) removing a suffix from the query; and (3) all other changes. As the search updates live when the user types a query, the first two categories are the most common, and in these cases the search does not need to be completely restarted.

When the query is appended (case 1), the search becomes more restrictive, and some of the existing search results might no longer be valid. Finder Highlights rechecks these existing results, then continues with the search using the new query; restarting the search is not required.

In the second case, a suffix is removed from the query. All existing results are still valid, however new results might now exist that were not previously valid. The search is restarted, including rechecking the Finder Highlights index, however existing results are preserved.

In the third case, there may be both existing results that are now invalid, and previously invalid results that are not valid. The search is therefore completely restarted.

Dynamically handling moved or renamed items

Users may potentially move or rename items during a search. When this occurs, changes to the results tree and search queue are required for each moved or renamed item.

The results tree is updated by removing the subtree rooted at the source location, and adding it again at the destination location. If the item has been renamed, the new name is checked to verify if it matches the search query. Any score changes, whether a result of location changes or renaming, are propagated up the hierarchy. If the moved or renamed item was not in the results tree, its name is checked, and if it is now a match, it is added.

After the results tree is updated, each item in the search queue is checked

to see if it is a descendant of the item, or the item itself, and its path is updated accordingly.

8.4 Field Evaluation of Finder Highlights

The primary objective of the research in this chapter is to learn how Icon Highlights and Search Directed Navigation are used in the real world. Finder Highlights addresses the interface and algorithm design issues described above, allowing for such an evaluation.

Finder Highlights was deployed in a four-week field study, logging details about Finder usage, Icon Highlights predictions and Search Directed Navigation searches. To protect privacy while maintaining an ability to identify revisitations, logs recorded hash codes of file paths, rather than the original paths. Logs were stored locally on participants' computers until the end of the study.

8.4.1 Procedure

Twenty OS X 10.8 users (4 female) participated in the study, aged between 23 and 66 (mean 32.0). One participant withdrew from the study due to the purchase of a new computer, leaving 19 submitting data. Self-reported daily computer use ranged from 1 to 14 hours per day (mean 7.2 hours). Participants were recruited from a range of sources: four were tertiary students, eight were university staff or faculty, and eight worked outside academia.

Communication with participants occurred online. After agreeing to an informed consent form, participants completed a pre-study survey, asking basic demographic information and general questions about how they retrieved files. They were then instructed to download the Finder Highlights software and to watch a three minute video that briefly explained the techniques. The video directed participants to use their computers as normal during the study, and to use the techniques as little, or as much, as they liked. After four weeks, participants were emailed with instructions to uninstall the software, submit their logs, and complete a post-study survey. The consent form and surveys are reproduced in Appendix C.

8.4.2 Log File Data Analysis

Ultimately, the techniques included in Finder Highlights are intended to reduce file retrieval times. However there are complexities in determining these times from interaction logs, including external distractions, cognitive efforts prior to task initiation, and the range of tasks possible within a file browser. In particular, certain assumptions were made to establish an objective time at which each retrieval began (in contrast, lab experiments can begin timing at the point an artificial stimuli is shown). In the analysis of retrieval times, time measurements were determined to begin at the later of (1) the time a browser window was opened; (2) the time of the first navigation event after either another retrieval, use of another browser window, or a file management task (e.g. moving or deleting an item); and (3) a navigation event that was more than 30 seconds after the previous one. Time measurements ended when a file was opened. Retrievals of items on the desktop were excluded, since Finder Highlights had no effect on the desktop interface.

To further characterise retrievals, *step times* were also analysed – that is, the time between opening successive folders while navigating to a file. To reduce noise, steps were excluded from the analysis if they included: an upwards step through the hierarchy; use of history buttons, sidebar links, search, or another window; a change in the view configuration; file management activities (e.g., moving a file); or if the step took longer than 30 seconds.

Baseline comparison between the unmodified Finder and Finder Highlights were performed by comparing logs of file retrieval from the characterisation study in Chapter 5 (hereafter referred to as *Finder logs*) with the logs generated by Finder Highlights. Logs from the Finder and Finder Highlights studies were analysed with exactly the same assumptions using exactly the same analysis tools. In particular, the assumptions for measuring retrieval and step times were identical. Though there were minor differences in the collection of the two datasets (i.e. Finder Highlights ran on a slightly newer version of OS X, and recruitment strategies differed slightly), these differences were minor and unlikely to significantly affect results. All times were log-transformed for ANOVA analyses to reduce the skew of the data.

Results incorporate both quantitative data retrieved from the logs and qualitative data from survey responses. Only the first 28 days of log data was included in analysis for each participant, though one participant only used the software for 20 days.

8.4.3 Retrieval Times and Step Times

Between-subjects analyses of retrieval and step times were conducted using Finder and Finder Highlights logs. The Finder logs included 1689 retrievals, while the Finder Highlights logs included 2521.

The mean retrieval time with Finder Highlights was 10.6 s (s.d. 3.4), 13% faster than the Finder mean of 12.2 s (s.d. 4.7): $F_{1,42} = 7.4, p < .01$. The mean for the Finder retrieval times is supported by a recent result from Bergman et al. [30], which reported a similar mean Mac OS X file retrieval time of 12.6 s. Of more interest, however, is the breakdown by technique, as Finder Highlights is primarily a mechanism by which to evaluate them. These results are provided below.

Icon Highlights

To examine the independent contribution of Icon Highlights, retrievals that involved search-based interfaces were excluded. Results were similar to the overall results above, which is best explained by the low proportion of retrievals using search or Search Directed Navigation (see below). Mean retrieval times with Icon Highlights were 9.8 s (s.d. 3.3), compared to 12.1 s (s.d. 4.8) without it – $F_{1,42} = 9.4, p < .005$. An analysis of the mean depth of retrieved files in both studies confirmed that this difference in retrieval times was not caused by differences between the participants' file structures, with no significant difference between retrieval depths in the two datasets. Further, a comparison of step times confirms that participants selected targets within the folder view more quickly when Icon Highlights were available (mean 3.9 s, s.d. 1.4) than when using the standard Finder (mean 4.5 s, s.d. 1.5) – $F_{1,42} = 4.2, p < .05$. Importantly, as Icon Highlights are intended to reduce visual search time by highlighting likely targets, this reduction provides strong evidence that this design intention is fulfilled. Both step and retrieval times for Icon Highlights are summarised in Figure 8.8.

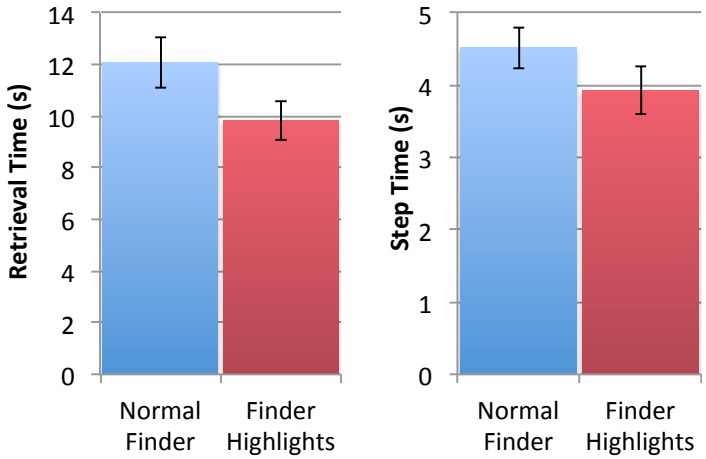


Figure 8.8: Non-search retrieval and step times with and without Finder Highlights, showing the effect of Icon Highlights. Error bars ±1 st. err.

Search Directed Navigation

In the earlier lab study (Chapter 7), Search Directed Navigation was highly efficient and extremely well received by participants. Results from the field study, however, are very different – not because Search Directed Navigation was inefficient or disliked, but rather because it was largely ignored. It was actively used by only 4 of the 19 participants, who used it in between 1% and 44% of navigational steps (mean 14%); these participants averaged 9.8 retrievals and 89 steps using Search Directed Navigation. This contrast between extremely positive lab study results and substantially neutral findings due to disuse in the field amplifies the motivation for this research, in migrating analysis from the lab to the real world.

The lack of data on active use of Search Directed Navigation complicates its rigorous analysis. Further, as Search Directed Navigation introduces new functionality – guiding users to files through the hierarchy in response to a search term – there are difficulties in determining an appropriate baseline for comparison. Limited insights can be gained by comparing steps times following Search Directed Navigation activation for the four participants who used it (mean 2.9 s, s.d. 1.3) to other steps made by the same participants (3.8 s, s.d 1.7), however statistical tests are inappropriate due to the above reasons.

Step Times with Correct and Incorrect Highlighting

Highlighting functionality is likely to assist users when their intended target is highlighted, but may distract when other items are highlighted and the target is not. To investigate this issue, all steps where Icon Highlights were available were categorised in one of three ways to produce an analysis factor *step type*: *positive highlights*, where the selected item was highlighted; *negative highlights*, where other items were highlighted and the selected item was not; and *no highlights*, where no items were highlighted.

A within-subjects analysis of variance showed a significant effect of *step type* – $F_{2,36} = 5.3, p < 0.01$. Step times were fastest with *positive highlights* (3.4 s), followed by *no highlights* (3.9 s), with *negative highlights* substantially slower (5.1 s). This result provides further evidence that Icon Highlights, when providing correct predictions, was successful in helping users quickly identify and select targets. While the result also suggests that highlighting items other than the target slows performance, this can at least partially be explained by varying folder sizes: folders with *negative highlights* contained many more items (mean 73.9) than either *positive highlights* (mean 21.8) or *no highlights* (mean 24.2). Therefore, both the slower performance and the incorrect highlighting can be partially attributed to the substantially larger folder sizes. Further research is necessary to accurately determine the negative impact of 'distractor' highlighting.

Further analysis of steps showed that 46.5% had *positive highlights*, 29.3% had *negative highlights*, and 24.2% had *no highlights*. In contrast to the 46.5% of selected items that were highlighted, an average of 22.5% of items were highlighted in each folder, suggesting that the AccessRank prediction algorithm provided good predictions of upcoming file selections. The average rank of a revisited item in its folder was 3.4 (18.2% of all steps opened a top-ranked item), while an average of 3.0 items were highlighted at each step.

8.4.4 Questionnaire Responses

After four weeks of use, participants completed a post-study questionnaire that gathered both quantitative subjective responses (using five-point Likert scales from 'strongly disagree' (1) to 'strongly agree' (5)) and participant

Statement (paraphrased)	Strongly Disagree (1)	Disagree (2)	Neutral (3)	Agree (4)	Strongly Agree (5)	Mean
Icon Highlights						
IH was useful	0	2	3	9	5	3.9
IH reduced cognitive load	0	3	7	5	4	3.5
Would like IH as permanent feature	0	4	3	8	4	3.6
Predictions seemed accurate	0	2	1	14	2	3.8
Could predict if target would be highlighted	0	1	4	8	5	3.9
Highlighting technique clear and intuitive	0	0	2	11	6	4.2
Search Directed Navigation						
SDN was useful	3	2	9	2	3	3.0
SDN affected file retrieval behaviour	2	2	10	2	3	3.1
Would like SDN as permanent feature	2	2	9	2	4	3.2
Results seemed accurate	0	0	4	2	4	4.0
SDN made it harder to use search	4	6	1	3	0	2.2
Confused between SDN and search	5	5	1	3	0	2.1
Highlighting technique clear and intuitive	0	0	2	4	6	4.3
General						
Hard to locate occluded highlights	3	8	2	2	1	2.4

Table 8.1: Participant agreement with statements about Icon Highlights (IH) and Search Directed Navigation (SDN).

comments. The quantitative responses are summarised in Table 8.1, which shows abbreviated forms of the Likert-scale questions asked.

Likert scale responses reflect the log results above. Participants responded more favourably to the question of whether Icon Highlights was useful (14 responses above neutral, 2 below) than for the equivalent question with Search Directed Navigation (5 above neutral, 5 below). Participants also felt that the Icon Highlights predictions were accurate (16 above neutral, 1 below) and predictable (13 above neutral, 1 below), and that its highlighting technique was clear and intuitive (17 above neutral, none below). The positive response for the predictability of highlighting is important, as prior research has shown that users can more quickly identify targets using pre-attentive search if they know in advance that they will be highlighted in a particular way [178].

Only three participants agreed that it was difficult to locate occluded highlights, such as those that require scrolling to see. This result implies that there is not a pressing need for features that emphasise their presence. Nevertheless, potential interfaces to achieve this are outlined in the discussion.

Icon Highlights

The survey also asked participants to rate the effect of Icon Highlights on retrieval times and error rates. Participants generally stated that it improved retrieval times (7 felt it made them faster, 1 slower), especially when retrieving previously visited files (15 faster, none slower), while it had a negligible effect for new or rarely visited files (1 faster, 1 slower). Icon Highlights also had a positive impact on perceived error rates (7 reported fewer navigation errors, none more).

The participants' general comments on Icon Highlights referred to the technique being most useful when navigating through many successive directories, when lots of items in a folder had similar names, and when revisiting specific items – particularly in large folders. However, a number of comments suggested possible refinements: (1) that too many items were highlighted towards the end of the study (one participant), which could be rectified by reducing the per-folder highlight limit or by considering the total number of items in a folder when calculating a limit; (2) that there was not enough difference between prominent and dim highlights (one participant), which could be rectified by calibrating the highlighting formula; (3) that the highlight colour was not visually appealing or should be customisable (six participants); (4) a lack of keyboard navigation, which would be useful to move between highlighted items (one participant); (5) confusing interaction between highlights and existing colour labels (one participant), which suggests the need for refining the colour blending methods used in Finder Highlights; and (6) that highlights could be unpredictable when opening items sequentially, where the highlighted items were not always the most recent items – this was caused by the lack of transitivity of AccessRank's stability component, causing highlights to sometimes be selected seemingly arbitrarily; this

effect could be reduced by calibrating AccessRank's parameters, for example by reducing the δ parameter.

The participants' comments also suggested a perception that the techniques might be most effective for users with disorganised file structures. *P8: "There is more and more mess on my hard drive, so it is useful to highlight files that I actually use".* Indeed, Icon Highlights may be of more benefit to 'pilers' than 'filers' (described in Chapter 3); those with disorganised hierarchies may be more reliant on visual search, perhaps due to larger folder sizes or less inherent knowledge of file locations.

One participant requested that the highlights should only appear when a modifier key is held down, although this could greatly reduce the method's benefits by changing it from a passive method (requiring no user input) to an active method (requiring explicit user action); as discussed below, Search Directed Navigation (an active method) was rarely used despite its potential benefits. Another participant suggested a feature to manually remove highlights from certain items, suggesting that they were periodically aware of the distraction caused by *negative highlights*.

While Icon Highlights were designed to help retrieve files, one participant said that its benefits were primarily in reducing *organisation* time. This participant had previously made extensive use of OS X's manual colour labelling feature, meaning that Icon Highlights provided no additional benefit at retrieval-time. However, he commented that it was a very useful automatic replacement for manual labelling. Another unexpected use, referred to in comments from four participants, was that sometimes the *absence* of highlights was particularly useful. For example, in ordered lists of movie files, the highlighted items would indicate which had already been watched, and the first unhighlighted item following them would correspond to the video they wanted to watch next. In another example, a participant sometimes knew a target item would not be highlighted, and could ignore highlights to speed up visual search.

Navigating with Icon Highlights normally occurred when participants already would have used navigation, however some participants stated that they occasionally switched from other methods to gain its advantages. This included two participants each in place of 'Open Recent' menus and search.

Search Directed Navigation

As expected, considering its relatively infrequent and light use, the dominant response for questions concerning use of Search Directed Navigation was neutral. However, as supported by the logs, a minority of the participants found it useful, with three expressing strong agreement regarding its usefulness. Four said they retrieved files more quickly overall because of SDN (four for previously visited files and one for new or rarely visited files), with it having no effect for the remainder, who did not use it.

Those who used it often used it in place of navigation, rather than just as a search substitute, and appreciated the context it provided. P5: *"Just loved it! Unlike the normal search, my current path stays the same, and I don't have a list of items coming from nowhere... I was comfortably guided through what I wanted. Sometimes [with Finder search] several files match the query and the results are all [shown together]. Here I can easily focus on the one file I was looking for. No ambiguities."*

P4 commented that Search Directed Navigation resulted in lower error rates than search, and facilitated learning of item locations, in line with its design goals: *"Sometimes I will use [Spotlight search], but if a file doesn't appear that I am looking for, I am no closer to finding it. With Search Directed Navigation... usually I also have a vague idea of where the file is, so I was able to use a starting top level folder as a starting point. This allowed me to usually find my file much more quickly, and with more success, but I also got the benefit of being reminded of where I stored the file and other files that I passed on the way."*

P6 also mentioned this property, commenting *"I really liked the way it seemed to teach the directory structure to me; traditional find would hide the structure from me"*, and referring to Icon Highlights as a way to retrieve files faster, but Search Directed Navigation as a way to remember his structure.

However, most participants rarely or never used Search Directed Navigation. Common reasons for this were that participants already knew where their files were (12 participants), do not use search much (8), forgot or did not think to use it (4), prefer traditional search interfaces (2), do not navigate to files much (2) or that it was too slow to find results (2). One participant

commented that he would have used SDN frequently had it supported navigation to highlighted items using the keyboard. Two participants did not realise that Search Directed Navigation only searched filenames, and were confused by this, with another suggesting that a content searching option would be useful.

A common theme in the explanations of the lack of use of Search Directed Navigation were comparisons with traditional search interfaces, which previous studies (as well as the findings in Chapter 5) suggest are often used as a method of last resort [19, 25, 32, 34]. Several participants used 'search' as a misnomer when referring to Search Directed Navigation in comments. These comparisons indicate that many participants did not consider Search Directed Navigation for tasks where they would not already use search, limiting its utility to those retrievals for which both traditional search and Search Directed Navigation were suitable. This issue is discussed further in the discussion.

Despite the low uptake, the strong positive reactions from those who used it suggest that Search Directed Navigation can be a valuable tool once familiarised with its use.

8.5 Design of Unimplemented Techniques and Features

Chapter 7 introduced three file retrieval interfaces: Icon Highlights, Search Directed Navigation and Hover Menus. Finder Highlights included implementations of Icon Highlights and Search Directed Navigation, but did not include Hover Menus. This section describes the design of a deployable version of Hover Menus, as well as the design of potential interfaces to indicate the presence of occluded highlights.

8.5.1 Hover Menus

Deployment of Hover Menus would require both a suitable interface design, as well as an efficient back-end algorithm to determine menu content. This section discusses potential interface designs for a Hover Menus-like technique, followed by a description of a potential algorithm that determines the contents of Hover Menus using a modified version of AccessRank. Except where

otherwise stated, AccessRank data is stored and updated in the same way as with Icon Highlights.

Alternative Interface Designs for Hover Menus

The user interface for Hover Menus described in chapter 7 could be used unmodified in a real world implementation. However, possible alternative designs are discussed below.

The Hover Menus design requires users to first find the folder corresponding to the next step of their retrieval. Alternate designs could omit this step, potentially increasing the benefits of using the shortcuts they provide, at the expense of lower accuracy. This could be accomplished, for example, by showing a list of suggestions in the sidebar, which are identical to the items that would be shown in the hover menu for the current folder.

Further opportunities arise when Search Directed Navigation is combined with Hover Menus. When Search Directed Navigation is active, the items in Hover Menus (or equivalent interface) could be restricted to items that would be highlighted by Search Directed Navigation. Alternatively, a popup could appear below the search field that shows these results directly, much like URL suggestions in a web browser, but with the added benefit of incorporating a benefit weighting based on hierarchy depth.

Benefit Weighted AccessRank – an Algorithm to Determine Menu Content

The original prototype of Finder Highlights used a hierarchy where all suggested folders were at a fixed depth, and all files were at a different fixed depth. This meant that items in Hover Menus – where files and folders are ranked separately – could be ranked purely based on AccessRank score. However, in a real file hierarchy, items are at varying depths. Using unmodified AccessRank scores would mean that the menus would be predominantly populated with shallow items, as folders generally have scores that are greater or equal to that of their descendants. However, the benefits of Hover Menus are greater when more hierarchy levels can be skipped.

To address this issue, Hover Menus can make use of a modified AccessRank score to determine item rankings, called Benefit Weighted AccessRank

(BWAR). Consider a hierarchy node N_1 at depth d_1, and one of its descendants, N_2, at depth d_2. Let the *benefit weighting* be the number of levels between these two nodes, i.e. $d_2 - d_1$. Further, let w_{N_2} be the AccessRank score of N_2, and $w_{BWAR}(N_1, N_2)$ be the BWAR score for N_2, relative to N_1. The latter can be calculated as:

$$w_{BWAR}(N_1, N_2) = (d_2 - d_1)w_{N_2}$$

The purpose of this modified score is to consider the total benefit of providing a shortcut to an item – that is, the likelihood of it being relevant, multiplied by the potential time saving that using the shortcut would provide. As found in Chapter 6, the original AccessRank score acts as a suitable approximation of the probability that an item will be accessed. Furthermore, by assuming that each step through the hierarchy takes constant time, multiplying by the number of levels between two locations provides a suitable method of approximating the potential time saving.

To implement this modified score calculation efficiently, all items with scores are represented in a tree structure. Each node also maintains a variable containing the depth of its subtree – i.e., the maximum distance between it and any of its descendants. The depth of the subtree of a leaf node is zero. Each node also stores its previous list of Hover Menus predictions, if applicable, where the list contains only those items actually displayed in the menu. References contained in this list should be updated if any of its items are moved, renamed or deleted.

When determining the items to appear in the Hover Menu for a folder, the folder becomes the Hover Menu's root node, N_1. Its descendants are then ranked based on their BWAR scores, relative to N_1. The order of items in N_1's previous prediction list is used by AccessRank's stability component during this ranking; note that when using AccessRank 2, the switching threshold is modified to be $T = w_{BWAR}(N_1, N_2)\delta$, instead of $T = w_{N_2}\delta$.

To avoid calculating BWAR scores and rankings for every item in the subtree, efficient culling can be used. First, recall that a node's CRF and Markov weights are both greater than or equal to those of any of its descendants, since each retrieval for a file also marks a retrieval for each of its

ancestors. Therefore, for versions of AccessRank that do not incorporate a time weighting (as in the Finder Highlights implementation of Icon Highlights), a node's AccessRank score will also be greater than or equal to the scores of its descendants. Using the subtree depth d_{max} that is stored for each node, the maximum possible BWAR score of any of the descendants of N_2, relative to the root node N_1, is therefore calculated as:

$$BWAR_{max}(N_1, N_2) = \left(d_2 - d_1 + d_{max_{N_2}}\right) w_{N_2}$$

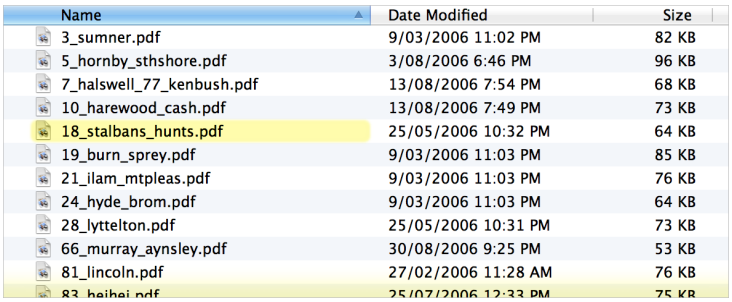
N_1's subtree can thus be searched using a priority queue, where a node N_2's priority is equal to $BWAR_{max}(N_1, N_2)$. When evaluating a node N_2, it is added to a set of candidate nodes only if $w_{BWAR}(N_1, N_2)+T > w_k$, where T is the AccessRank switching threshold, k is the number of results shown in the Hover Menu, and w_k is the BWAR score of the kth highest candidate score, or 0 if fewer than k candidates exist. Additionally, N_2's children, denoted by N_{2_i}, are added to the priority queue only when $BWAR_{max}(N_1, N_{2_i})+T > w_k$. The search ends when either the priority queue is empty, or the highest priority in the queue is less than the current value of w_k (which can increase over the course of the search).

Note that separate sets of candidate nodes will typically be maintained for files and folders, however for simplicity, they are referred to here as a single set.

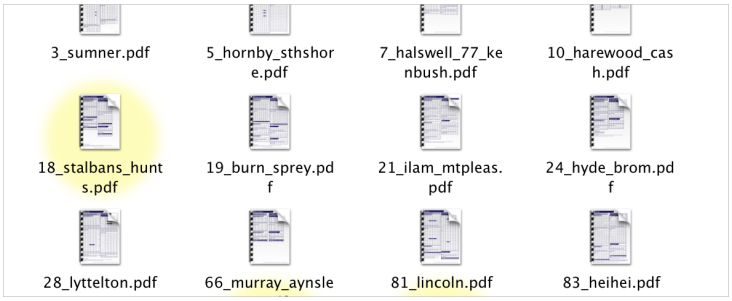
Once a candidate set is finalised, its items are ranked using AccessRank. The top k results are stored as the node's previous list of predictions and displayed in the menu.

8.5.2 Highlight Edge Indicators

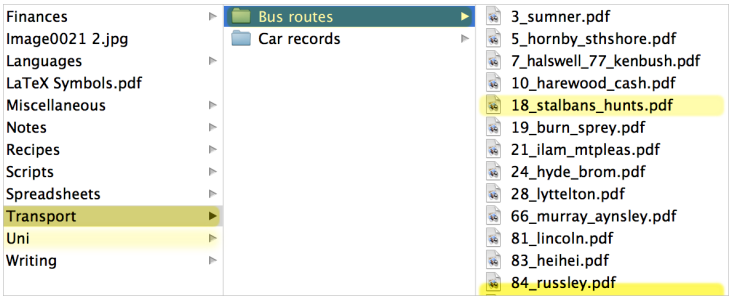
A limitation of visually highlighting items is that the prominence or existence of obscured highlights is not known, such as when viewing a large folder where scrolling is required. While the original design of Icon Highlights included scrollbar marks to indicate the locations of items outside the current view area, not all systems include scrollbars that are always visible. Furthermore, in two dimensional views such as icon view, locations cannot easily be deduced purely from markings in one dimensional horizontal and



(a) Whole edge highlighting



(b) Sticky item highlighting



(c) Stacked sticky item highlights

Figure 8.9: Edge highlight designs for Icon Highlights

vertical scrollbars. Although most participants in the Finder Highlights evaluation did not indicate that the need to emphasise obscured highlights was a pressing issue, this section describes potential methods to assist those who would like such a feature.

Techniques such as Halo [20] and Wedge [88] address the issue of occluded objects by using visualisations at the screen edge that imply the exact location of the object. Halo achieves this by displaying the edge of a ring, where the centre of the ring corresponds to the object, and Wedge further devel-

ops the idea by displaying non-overlapping wedges. In a file browser, precise two dimensional locations are less important than in other contexts, such as landmarks on a map. For example, horizontal location is of no relevance in list and column views, and there is typically minimal horizontal scrolling required in icon view. As such, simpler techniques than Halo and Wedge are likely sufficient.

Figure 8.9 shows two possible designs. The simpler, *whole edge highlighting*, is shown in Figure 8.9a. Whole edge highlighting shows highlights at the edge of the view, spanning the entire edge and indicating the direction of occluded highlights. In the example, the highlight at the bottom of the view indicates that highlighted items exist further down the view. Multiple edges could be highlighted if highlights appear in multiple directions, and corners could have diagonal sloped highlights to indicate any items that are outside the visible bounds of the view in two dimensions. The intensity of the highlight could be used to indicate the most intense occluded highlight, the number of occluded highlights, or the distance to the next occluded highlight.

Figure 8.9b shows *sticky item highlighting*, where all occluded highlights are shifted horizontally or vertically so that they are visible at the edge of the visible area. Items that are outside the bounds in two dimensions are shifted both horizontally and vertically. Figure 8.9c also shows sticky item highlighting, with a much brighter edge highlight. This shows multiple highlights 'stacked' (drawn on top of each other), and can be used as an indicator of the number of occluded highlights. As the user scrolls down and fewer highlights are occluded below the view, the edge indicator becomes dimmer.

While the above designs are shown for Icon Highlights, they would work equally well for Search Directed Navigation, with the exception that stacked highlights would not be apparent as the drawing method is not cumulative.

8.6 Discussion

To briefly summarise, conducting a real world evaluation of Icon Highlights and Search Directed Navigation required addressing several user interface design challenges in extending the previously reported research prototypes into

viable file browser features. It also required several extensions to existing algorithms. Results of a four-week field study in which real users employed Finder Highlights for their everyday file retrieval activities showed that Icon Highlights reduced file retrieval times. Search Directed Navigation was ignored by most users, but generated enthusiastic responses from those who used it.

8.6.1 Methodology – Lab versus Field

Step and retrieval times from the field study were in line with those from the original lab study, confirming the techniques' performance benefits in real world tasks. The main difference between the studies was the difference in usage of Search Directed Navigation: in the lab, it caused marked performance improvement and strongly positive subjective assessments, but in the field, it was largely ignored. It is likely that the reason there was a large difference between studies for Search Directed Navigation, but not for Icon Highlights, stems from the difference between *active* and *passive* techniques; lab studies are potentially more engaging than field studies, with implicit expectations for participants to use new or novel interfaces, meaning that lab studies overestimate the usage rate of active interfaces. On the other hand, users have no choice in whether they use passive interfaces, so gain the benefits in both contexts.

The field study also identified new use cases not identified in the lab study, such as the use of Icon Highlights as a substitute for manual highlighting, or to determine progress through a list of video files. This reinforces the view of Hertzum [96] that lab studies are biased towards *how* tasks are performed, rather than *what* tasks can be performed.

The findings from the field study underline and reiterate methodological concerns previously identified in several HCI studies [43, 60, 146]. The need for cross validation across multiple methods is well known (e.g., [96]), but it remains relatively uncommon in HCI. There are good reasons for not advancing beyond lab studies – the *potential* of innovative interface concepts can be established quickly and with strong rigour in controlled experimental studies; advancing beyond lab studies requires arduous and refined system

engineering; and longitudinal evaluation is difficult to conduct, not least because of the difficulty of finding willing participants who will make prolonged commitments to experimental software. Although these disincentives to the evaluation of real systems are substantial, the results in this study exemplify both the limits of lab studies and the value of field studies.

8.6.2 Overcoming Barriers to Adoption

The polarised results for Search Directed Navigation raise important and general challenges for interface designers. In the lab study performance with Search Directed Navigation was excellent. In the field study, at least some who used it loved it. But, most participants ignored it.

It seems likely that Search Directed Navigation can improve user performance in file retrieval tasks, as suggested by the lab study and the enthusiasm of those who used it. However, until users have their first experience of its benefits, they have little reason to try to exploit it, as doing so requires an overhead in learning and adapting to its new functionality.

There was also evidence that many participants hesitated to use Search Directed Navigation because of strong associations with search. In particular, all participants who agreed it was useful described it as a combination of search and navigation, while all three participants who described it as purely a search feature rated it negatively or neutrally. Furthermore, eight participants provided "I don't use search much" as a reason for not using Search Directed Navigation. While Search Directed Navigation was designed, in part, to bridge the gap between search and navigation, and thus to be used for a wider range of retrievals than search (which is typically used as a last resort [19, 25, 32, 34]), strong associations with search may have impacted its uptake. Perhaps framing Search Directed Navigation as less of a search-based technique would change perceptions and increase adoption – for example, by changing its name and removing its integration with the standard search field, instead using an alternate activation mechanism.

The barriers to adopting new and ultimately more efficient interface methods were also examined by Scarr et al. [165]. There are abundant opportunities for further research on methods that help users identify and capitalise

on unused interface methods that will ultimately assist them in becoming more efficient with computing tools, with 'Skillometers' providing one recent example [126].

8.6.3 Predictive Highlighting More Broadly

Icon Highlights and Search Directed Navigation both use highlighting methods to draw the user's attention to anticipated targets for retrieval. Although the specific application area for Icon Highlights and Search Directed Navigation is the important and frequent task of file retrieval, the results have implications for many other application areas. Considerable previous research has explored predictive highlighting (e.g., [5, 68, 134, 142]), with the predictions often based on simple models of most frequent or most recent accesses. As predictive algorithms become more accurate, there is likely to be greater success in migrating related techniques into production systems. Before this occurs, there are fruitful research opportunities in examining the design space and effectiveness of different highlighting schemes.

8.6.4 Prediction Versus Interaction

Icon Highlights and Search Directed Navigation use sophisticated algorithms to predict the user's upcoming actions. Their predictive algorithms are currently exploited to assist the user's manual methods for file retrieval – they help users identify targets and learn the navigational actions required so that users can improve performance over time. However, there is an interface design tension between the accuracy of the predictive algorithm and the interface methods used to present those predictions. For example, if AccessRank or any other algorithm had near perfect prediction, then it would be near pointless to require the user to navigate to the file, and instead it could be autonomously opened by the system. There are interesting theoretical and design challenges in understanding the relative merits of different interface methods in the presence of improving predictive algorithms, with some of these issues discussed in [75].

8.6.5 Purpose of Stability in Predictions

AccessRank emphasises the need for prediction stability to enable greater consistency in user interfaces. Its true importance relates to the benefits of users being able to predict in advance where items will be or what items will be emphasised [178]. Quantitative measures used to assess stability are normally a suitable approximation of this property, however there are clear cases where a mismatch occurs – for example, when accessing video files sequentially, when the user expects the most recent files to be highlighted, rather than a temporally stable set of items. This highlights the need for further work on the development of new metrics or techniques to better assess how well algorithm predictions match user expectations.

8.6.6 Highlighting Aesthetics and Effectiveness

Findlater et al. [68] commented that "*it may be difficult to design visually attractive real world interfaces that use highlighting, and even harder to add colour highlighting to an existing interface*", since such highlighting would compete with other uses of colour in these interfaces. The implementation of Icon Highlights and Search Directed Navigation, and the positive survey responses to the highlighting mechanisms, indicate that it is achievable in the context of file browsers. Findlater et al.'s comment came in response to a study by Gajos et al. [74], in which a highlighting technique for command access was unsuccessful.

For highlighting techniques to be successful, it is likely that the following requirements must be met: (1) that the underlying interfaces are not spatially stable, meaning that spatial memory alone cannot be used to remember item locations; and (2) that users can anticipate whether their target will be highlighted [178]. While both of these are true for Icon Highlights and Search Directed Navigation, this was not the case in Gajos et al.'s study.

8.7 Conclusion

Finder Highlights is a robust and real software system that augments the OS X file browser, the Finder, with techniques designed to improve navigation-

based file retrieval. It was designed and implemented in order to test Icon Highlights and Search Directed Navigation, two techniques that earlier lab studies showed to have strong potential for improving file retrieval. This chapter presented the interface and algorithmic design challenges associated with advancing from the lab to the field. The algorithmic advances, in particular, represent research contributions that can be reused by future researchers working on interfaces that predict upcoming user actions.

A four-week field study of Finder Highlights showed marked differences between the lab and field results. While the field study results show that the potential of Icon Highlights identified in the prior study is realised in the field, the lab study results for Search Directed Navigation did not fully translate to the field. The findings have clear implications for the design of future file browser interfaces, which should incorporate features similar to those of Finder Highlights, and for HCI experimental methodology. This work will hopefully inspire more novel interactive techniques to be tested in the field as well as the lab.

Part V

Discussion, Further Work and Conclusions

Chapter IX

Discussion and Further Work

File retrieval is a common and important task for computer users. The work presented in this thesis aims to improve user interfaces for file retrieval: it described an empirical characterisation of file retrieval behaviour, introduced a new prediction algorithm that can be used in a variety of user interfaces, and proposed and evaluated three improvements to file navigation.

This chapter takes a broader look at the context of this thesis. It describes progress on research objectives, discusses the ways in which the research can be generalised to other areas, and provides direction for future research.

9.1 Progress on Research Objectives

The introduction described four goals for the thesis, further broken down from the overarching goal of *understanding and improving personal file retrieval*. These goals were:

1. Understand the methods with which users can currently retrieve files, and understand retrieval behaviour in the context of existing knowledge about file management.

2. Form an empirical characterisation of file retrieval behaviour based on how users retrieve their files in a natural setting.

3. Develop and evaluate a prediction algorithm that can accurately predict future user actions for use in user interfaces.

4. Design, implement and evaluate improved file retrieval methods that allow users to access their files more efficiently.

The first part of goal 1 was advanced with a description and categorisation of existing retrieval methods in Chapter 2. Chapter 3 further explored file retrieval in the context of retrieval across different domains, and provided a broad overview of existing literature on file management, addressing the remainder of goal 1.

Goal 2 was progressed with the empirical characterisation of file retrieval in Chapter 5. This characterisation was formed by conducting a four-week longitudinal study in which FileMonitor, a logging tool described in Chapter 4, was deployed to 26 participants. The characterisation provided a thorough description of file retrieval behaviour, including both quantitative measures such as file retrieval times and window reuse behaviour, and qualitative data from participant interviews.

Goal 3 was achieved with the design and evaluation of AccessRank in Chapter 6. Simulations were performed that showed that AccessRank outperformed existing algorithms over a range of datasets when considering the combination of both accuracy and stability of results. AccessRank was further developed in Chapter 8, where improvements were made to facilitate predictions in hierarchical file structures.

Goal 4 was advanced with the design and evaluation of Icon Highlights, Hover Menus and Search Directed Navigation. The techniques were initially described and evaluated in Chapter 7. This evaluation showed that they improved file retrieval times and were subjectively preferred over a standard navigation interface in a lab setting, and detailed the conditions under which each interface performed best. Chapter 8 extended this work by implementing fully functional versions of Icon Highlights and Search Directed Navigation as part of a plugin to the OS X Finder called Finder Highlights. A four-week longitudinal field study of Finder Highlights confirmed the benefits of these techniques in a natural setting, while highlighting limitations in the adoption of Search Directed Navigation outside a lab context.

Together, the progress on these goals significantly contributes towards the research community's knowledge of file retrieval, both in terms of *understanding* file retrieval (goals 1 and 2) and *improving* current interfaces (goals 3 and 4).

9.2 Context of This Thesis

This thesis provided new understanding of file retrieval and ways to improve it. However, it is by no means exhaustive, and there are many aspects of file management that were not investigated as part of this research.

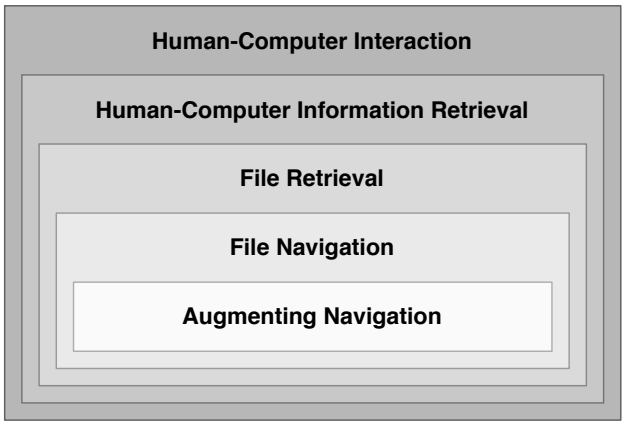


Figure 9.1: Research context of this thesis

Figure 9.1 provides a simplified diagram of the context of this thesis. Within Human-Computer Interaction, the thesis focused on *Human-Computer Information Retrieval* [130]. This excluded other aspects of file management, such as file organisation; although aspects of this were interweaved throughout the thesis, it was not a primary focus. Within Human-Computer Information Retrieval, the focus was on *file retrieval*, as opposed to other domains such as retrieval of emails or previously visited websites. The improved file retrieval interfaces described in Chapters 7 and 8 then further focused on *file navigation*, having found in earlier chapters that navigation was the most common file retrieval method. However, there are undoubtedly countless improvements that could be made to other retrieval methods, as well as the design of completely new methods. Within file navigation, the focus was on *augmenting current navigation interfaces*, adding additional features rather than modifying existing ones. Methods that are still navigation-based, but challenge the underlying assumptions of existing navigation interfaces, were not explored.

Many of the results in this thesis generalise to some of the areas not

explicitly explored, in ways described in the following section. However, there remain abundant research opportunities in the domain of file management that were outside the scope of this thesis.

9.3 Research Generalisability

The focus on this thesis was on understanding file retrieval and improving it through the use of particular enhancements to file navigation. However, aspects of this research are generalisable and also apply to other areas. This section focuses on three such areas, while the discussion sections of individual chapters provide additional insights.

9.3.1 Characterisation of File Retrieval

The characterisation of file retrieval behaviour in Chapter 5 provided a broad and detailed description of how users retrieve their files and the issues they face. While this characterisation was important in advising the design of the file retrieval improvements discussed later in the thesis, they are undoubtedly also relevant to the design of other improved retrieval methods.

The results also have broader implications on research methodology, both for studies of file retrieval and of user interfaces in general. Lab studies often consider tasks in isolation (e.g., [29]), which facilitates robust statistical analysis. However, the significant amount of window reuse in file navigation shows that this is not how users perform tasks in practice; they multitask and make use of existing resources that are available to them from previous tasks. This has significant implications on the external validity of studies that do not consider this, and suggests that studies that do not 'reset' the experimental interface after each task may be more reflective of real-world usage.

9.3.2 AccessRank

AccessRank was designed to be domain independent – that is, it does not rely on the attributes of any particular domain, using only information about the times of previous item accesses to make its predictions. While it was used

in the context of file retrieval interfaces in this thesis, it would be suitable to many other domains, particularly those it was evaluated on: commands, websites and window switches.

In many domains, AccessRank can be used unmodified. Those that use hierarchical data can incorporate the AccessRank enhancements described in Chapter 8. When global predictions are required across an entire hierarchy or sub-hierarchy, Benefit Weighted AccessRank (Chapter 8) can produce predictions that take into account the relative costs of retrieving items at varying hierarchy depths. While domain-specific enhancements are possible in some domains (for example, by incorporating the effect of different access methods for URLs), AccessRank can be used as-is to provide accurate and stable predictions with a range of domains and predictive interfaces. Chapter 6 provided some examples of interfaces into which AccessRank could be incorporated.

9.3.3 Predictive Interfaces

Icon Highlights and Search Directed Navigation showed that predictive interfaces, and in particular those that highlight likely items, can be highly effective. Highlighting could therefore be adopted in a wider range of interfaces. However, predictive interfaces must be designed with care – in particular, it is important that they retain a degree of spatial consistency in order to facilitate the transition to expert performance.

The specific interfaces introduced in this thesis could also be adapted for other domains. For example, Icon Highlights could be adapted to highlight links on web pages that are most likely to be of interest, determined based on both the user's access history and communal access patterns. Search Directed Navigation could also highlight links, in order to help train people how to access certain parts of large websites without the use of search. Similarly, the techniques could be adopted for retrieval of email messages – as has been suggested by Elsweiler et al. [64].

9.4 Future Work

While there is significant room for iterative refinement of the work described in this thesis, this section focuses on some wider issues that have growing importance as the world becomes more interconnected and technology becomes more ubiquitous. These include the increasing use of shared data as part of project collaboration, the use of multiple devices by single users, and changing file management paradigms as a result of greater security requirements.

9.4.1 Shared Data

The focus of this thesis was on retrieving items from personal file collections. These personal collections differ from shared repositories, in that the owner has sole control over, and has much greater familiarity with, their hierarchy structure. On the other hand, shared repositories – commonly found on the web – mean that only one person is responsible for the organisation of the information, while other users can focus on retrieval.

Tools such as Dropbox [59] blur this boundary. Dropbox integrates with file managers to provide a section of the file hierarchy which is shared with other users. While all users who have access to a particular part of this repository can modify its internal structure, these changes are synchronised, and any changes are imposed on other users.

Systems like Dropbox introduce interesting directions for future research. While the organisational load is shared amongst users, the loss of control is disadvantageous; in the file retrieval characterisation study (Chapter 5) one participant described this aspect as "really awkward". Karlson et al. [113] also describe this problem, naming it the *'Ownership Problem'*. There are abundant research opportunities in investigating ways to mitigate this issue – for example, how such tools could allow users to individually customise their view of shared data to accommodate their own organisational structure, while retaining the ability to synchronous data between users. Such customisations could also allow these shared files to be distributed amongst existing organisational structures, resolving the conflict between organising based on related content (e.g., by project) and based on purpose (e.g., for sharing) [57] that these tools currently introduce.

The reduced need to spend time organising these shared repositories must also be balanced with the greater difficulty retrieving files due to lower hierarchy familiarity. Greater adoption of shared repositories increases the importance of efficient retrieval methods, with those that can predict retrievals potentially playing an important role. While AccessRank (Chapter 6) makes predictions based on the access history of a single user, there is scope for improved algorithms that incorporate usage history from multiple users when acting on shared repositories. While this is common on the web (e.g., [36, 141, 76]), web prediction algorithms incorporate data from large numbers of users that allows for sophisticated analysis and prediction. Hybrid algorithms, incorporating communal access history for repositories shared with a small number of people, but biased towards the current user's history, are worthy of investigation.

9.4.2 Multiple Devices

While the previous section considered the problem of *one hierarchy to many users*, this section considers the opposite problem: *one user to many hierarchies*. That is, the effect on predictions and their presentation when a single user uses the same file on multiple devices.

The simplest case of this problem is when users have identical copies of their file hierarchy on multiple devices, with the content synchronised between them. Currently, the interfaces implemented in Finder Highlights (Chapter 8) consider only those past retrievals performed on the same device, and synchronisation tools would not be able to merge the prediction databases across devices. More sophisticated implementations of the retrieval techniques could add explicit support for considering retrieval history across multiple devices when making predictions, with the two main areas of work being the merging of prediction databases when synchronising devices, and considering the relative predictive power of accesses on each device on future accesses on the current device. Algorithmic enhancements would be required to support this latter consideration, in order to efficiently make predictions based on past accesses where the device used for each access must be considered, and this might involve different (but compatible) representations of

the prediction database on each device.

In other cases, files or file hierarchies might be *similar* across devices, but not identical. This could occur, for example, if a folder is copied to another device but is not subsequently synchronised. Alternatively, it could occur when synchronising with a device which uses a different representation to store content – for example, synchronising between a desktop computer, with a full file hierarchy, and a mobile device, with more limited support for hierarchical file structures. In these cases, the individual files still exist on multiple devices, but they are not necessarily organised in the same form. Accessing one of these files on one device, however, may still impact the probability that it will be accessed on a different device, and future predictive algorithms and interfaces could add support for tracking files across devices independent of hierarchical structure.

9.4.3 Changing Paradigms

As security concerns grow, new file management paradigms are emerging. As an example, iOS uses application sandboxing, restricting applications to their own storage areas. Apple's iCloud technology [9] allows users to seamlessly synchronise files across devices (for example, an iOS device and a desktop computer), however these files are stored in *ubiquity containers* – sandboxed areas of the file system reserved for a specific application or file type. Each ubiquity container contains a 'Documents' folder, the contents of which is presented to the user when they wish to save a file to, or open a file from, the application's iCloud storage area. However, there is no direct support for creating additional subfolders inside these Documents folders.

While this approach eases the organisational load by automatically filing based on application, it presents several problems. First, it does not allow users to create their own hierarchies within each ubiquity container, negating many of the benefits that hierarchies provide when working with large numbers of files. Second, it forces users to organise content in undesirable ways. By requiring that users save content in iCloud's storage area in order to synchronise it across devices, it forces users to organise based on *purpose* [57], when they more often consider their content based on *project* or *role*

[23]. Furthermore, within iCloud, it forces users to organise content based on *format*. These two aspects create a mismatch between users' mental models of their files and how the computer presents them, forcing users to change their thinking when retrieving a file to first consider attributes that are less important to them, such as whether it is a synchronised document or what format it is.

Unless improved non-hierarchical retrieval methods are developed, it is unlikely that file management paradigms such as those used by iCloud will become ubiquitous. While they are functional for smaller numbers of documents, the inability for users to organise based on project or role make them infeasible for large numbers of documents, as is common in modern computing. Further research is therefore needed to develop methods that allow users to use organisational schemes consistent with their mental models, while also accommodating changing security requirements and the different synchronisation, sharing and backup requirements of different files. Some of these issues are also explored in Placeless Documents [57].

Chapter X

Conclusion

Personal file retrieval is a common, but tedious task for all computer users. This thesis was motivated both by the knowledge gap faced by researchers in how users retrieve their files, and by the desire to create improved retrieval interfaces. It addressed these motivations in three parts. First, with an empirical characterisation of personal file retrieval, describing in detail the files people retrieve and the methods they use to do so in everyday work environments. Second, with a new prediction algorithm, AccessRank, for use in predictive user interfaces. Finally, with three enhancements to navigation-based file retrieval which were found to improve retrieval times in both structured lab studies and everyday retrieval tasks as part of a four-week field study.

This thesis made six primary research contributions, relating mainly to the domain of personal file retrieval, but in some cases with broader application:

1. A review of file retrieval methods and file management. This review allows other researchers to more quickly evaluate currently retrieval methods and opportunities for further research.

2. The development of FileMonitor, a tool that logs file retrieval behaviour on OS X. FileMonitor enables accurate and automated collection of large amounts of file retrieval data, which can be used either to characterise file retrieval behaviour or for evaluating file retrieval interfaces or prediction algorithms. The description of FileMonitor's implementation is also of use to researchers developing similar tools.

3. An empirical characterisation of file retrieval behaviour. A four-week longitudinal study involved 26 participants running FileMonitor on

their personal computers, followed by 30 minutes interviews about their file management behaviours. FileMonitor logs and interview responses provided a detailed characterisation of file retrieval behaviour, including specific analyses of revisitation, window reuse, and the use of different retrieval methods. Findings from this characterisation provide insights into both the design and evaluation of file retrieval tools.

4. The development and evaluation of AccessRank, a prediction algorithm designed for use in a range of user interfaces. AccessRank was designed with both accuracy and stability of results in mind. Simulations on a range of datasets compared various configurations of AccessRank with existing algorithms, showing that AccessRank outperformed existing algorithms when considering the combination of accuracy and stability, and also demonstrating the strengths of different algorithms across different datasets.

5. The design of three interfaces that augment a standard file browser to aid file navigation tasks: Icon Highlights, which highlight items that are most likely to be accessed next; Search Directed Navigation, which highlights items that match, or contain items that match a filename query; and Hover Menus, which provide a menu containing shortcuts to likely files and folders deeper in the hierarchy. A lab study of 16 participants showed that each improved file retrieval performance and was subjectively preferred over a standard browser.

6. The design and evaluation of fully functional implementations of Icon Highlights and Search Directed Navigation, implemented as part of Finder Highlights, a plugin to a real-world file browser. The implementations include design and algorithmic enhancements to enable the techniques to work in a real-world setting, including a specialised version of AccessRank to predict future file retrievals for Icon Highlights, and a search algorithm to determine filename matches for Search Directed Navigation. Finder Highlights was evaluated in a four-week field study with 19 participants, with results confirming the performance

benefits provided by Icon Highlights and highlighting the advantages that field studies can offer over lab studies.

This research provides the foundation for further work in several areas. First, the empirical characterisation of file retrieval highlights several areas in which file retrieval tools can be improved. Second, many user interfaces could benefit from incorporating AccessRank predictions to improve prediction accuracy as well as results stability. Third, the improved navigation techniques described in this thesis could be further improved based on feedback from field study participants.

References

[1] Abrams, D., Baecker, R., and Chignell, M. Information archiving with bookmarks: personal web space construction and organization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, ACM (New York, NY, USA, 1998), 41–48.

[2] Aceituno, J., and Roussel, N. The Hotkey Palette: Flexible Contextual Retrieval of Chosen Documents and Windows. Tech. Rep. RR-8313, INRIA, June 2013.

[3] Adar, E., Teevan, J., and Dumais, S. T. Resonance on the web: web dynamics and revisitation patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 1381–1390.

[4] Agarawala, A., and Balakrishnan, R. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, ACM (New York, NY, USA, 2006), 1283–1292.

[5] Alexander, J., Cockburn, A., Fitchett, S., Gutwin, C., and Greenberg, S. Revisiting read wear: analysis, design, and evaluation of a footprints scrollbar. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, ACM (New York, NY, USA, 2009), 1665–1674.

[6] Alexander, J., Cockburn, A., and Lobb, R. Appmonitor: A tool for recording user actions in unmodified windows applications. *Behavior Research Methods 40*, 2 (2008), 413–421.

[7] Apple Developer Connection. Working with Spotlight. `https://developer.apple.com/library/mac/#documentation/Car`

bon/Conceptual/MetadataIntro/MetadataIntro.html, retrieved 2013.

[8] Apple Inc. The Mac OS X Accessibility Protocol. https://developer.apple.com/library/mac/#documentation/Accessibility/Conceptual/AccessibilityMacOSX/OSXAXModel/OSXAXmodel.html, 2013.

[9] Apple Inc. iCloud Design Guide: iCloud Fundamentals. https://developer.apple.com/library/mac/documentation/General/Conceptual/iCloudDesignGuide/Chapters/iCloudFundametals.html, retrieved 2013.

[10] Apple Inc. Safari. http://www.apple.com/safari/, retrieved 2013.

[11] Arcuri, M., Coon, T., Johnson, J., Manning, A., and van Tilburg, M. Adaptive menus, Sept. 19 2000. US Patent 6,121,968.

[12] Arlitt, M., Friedrich, R., and Jin, T. Performance evaluation of Web proxy cache replacement policies. *Performance Evaluation 39*, 1-4 (2000), 149–164.

[13] Aula, A., Jhaveri, N., and Käki, M. Information search and re-access strategies of experienced web users. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, ACM (New York, NY, USA, 2005), 583–592.

[14] Azagury, A., Factor, M. E., Maarek, Y. S., and Mandler, B. A novel navigation paradigm for xml repositories. *Journal of the American Society for Information Science and Technology 53*, 6 (2002), 515–525.

[15] Balakrishnan, R. "beating" fitts' law: virtual enhancements for pointing facilitation. *Int. Journal of Human-Computer Studies* (2004), 857–874.

[16] Bälter, O. Keystroke level analysis of email message organization. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '00, ACM (New York, NY, USA, 2000), 105–112.

[17] Bao, X., Herlocker, J. L., and Dietterich, T. G. Fewer clicks and less frustration: Reducing the cost of reaching the right folder. In *Proceedings of the 11th international conference on Intelligent user interfaces*, ACM (2006), 178–185.

[18] Barreau, D. Context as a factor in personal information management systems. *Journal of the American Society for Information Science 46*, 5 (1995), 327–339.

[19] Barreau, D., and Nardi, B. A. Finding and reminding: file organization from the desktop. *SIGCHI Bull. 27*, 3 (July 1995), 39–43.

[20] Baudisch, P., and Rosenholtz, R. Halo: a technique for visualizing off-screen objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM (New York, NY, USA, 2003), 481–488.

[21] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I., and Grinter, R. E. Quality versus quantity: E-mail-centric task management and its relation with overload. *Human-computer interaction 20*, 1 (2005), 89–138.

[22] Bergman, O., Beyth-Marom, R., and Nachmias, R. The user-subjective approach to personal information management systems. *Journal of the American Society for Information Science and Technology 54*, 9 (2003), 872–878.

[23] Bergman, O., Beyth-Marom, R., and Nachmias, R. The project fragmentation problem in personal information management. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, ACM (New York, NY, USA, 2006), 271–274.

[24] Bergman, O., Beyth-Marom, R., and Nachmias, R. The user-subjective approach to personal information management systems design: Evidence and implementations. *Journal of the American Society for Information Science and Technology 59*, 2 (2008), 235–246.

[25] Bergman, O., Beyth-Marom, R., Nachmias, R., Gradovitch, N., and Whittaker, S. Improved search engines and navigation preference in personal information management. *ACM Trans. Inf. Syst. 26*, 4 (2008), 1–24.

[26] Bergman, O., Gradovitch, N., Bar-Ilan, J., and Beyth-Marom, R. Folder versus tag preference in personal information management. *Journal of the American Society for Information Science and Technology* (2013).

[27] Bergman, O., Tene-Rubinstein, M., and Shalom, J. The use of attention resources in navigation versus search. *Personal and Ubiquitous Computing 17*, 3 (2013), 583–590.

[28] Bergman, O., Tucker, S., Beyth-Marom, R., Cutrell, E., and Whittaker, S. It's not that important: demoting personal information of low subjective importance using grayarea. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 269–278.

[29] Bergman, O., Whittaker, S., Sanderson, M., Nachmias, R., and Ramamoorthy, A. The effect of folder structure on personal file navigation. *Journal of the American Society for Information Science and Technology (JASIST) 61*, 12 (2011), 2300–2310.

[30] Bergman, O., Whittaker, S., Sanderson, M., Nachmias, R., and Ramamoorthy, A. How do we find personal files?: the effect of os, presentation & depth on file navigation. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, ACM (New York, NY, USA, 2012), 2977–2980.

[31] Blacktree. Quicksilver. `http://www.blacktree.com/`, retrieved 2013.

[32] Blanc-Brude, T., and Scapin, D. L. What do people recall about their documents?: implications for desktop search tools. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, ACM (New York, NY, USA, 2007), 102–111.

[33] Boardman, R. Multiple hierarchies in user workspace. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, ACM (New York, NY, USA, 2001), 403–404.

[34] 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 the SIGCHI conference on Human factors in computing systems*, CHI '04, ACM (New York, NY, USA, 2004), 583–590.

[35] Boardman, R., Spence, R., and Sasse, M. A. Too many hierarchies? the daily struggle for control of the workspace. In *Proc. HCI International 2003*, vol. 1 (2003), 616–620.

[36] Bollen, J., Vandesompel, H., and Rocha, L. Mining associative relations from website logs and their application to context-dependent retrieval using spreading activation. In *Workshop on Organizing Web Space (WOWS), ACM Digital Libraries 99, August 1999, Berkeley, California*, Citeseer (1999).

[37] Bush, V. As We May Think. *Atlantic Monthly* (1945).

[38] Byrne, M. D., John, B. E., Wehrle, N. S., and Crow, D. C. The tangled web we wove: a taskonomy of www use. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 1999), 544–551.

[39] Capra III, R. G., and Pérez-Quiñones, M. A. Using web search engines to find and refind information. *Computer 38* (2005), 36–42.

[40] Carroll, J. M. Creative names for personal files in an interactive computing environment. *International Journal of Man-Machine Studies 16*, 4 (1982), 405–438.

[41] Case, D. O. Collection and organization of written information by social scientists and humanists: a review and exploratory study. *Journal of Information Science 12*, 3 (1986), 97–104.

[42] Catledge, L., and Pitkow, J. Characterizing browsing strategies in the World-Wide Web. *Computer Networks and ISDN systems 27*, 6 (1995), 1065–1073.

[43] Chiasson, S., Biddle, R., and van Oorschot, P. C. A second look at the usability of click-based graphical passwords. In *Proceedings of the 3rd symposium on Usable privacy and security*, SOUPS '07, ACM (New York, NY, USA, 2007), 1–12.

[44] Civan, A., Jones, W., Klasnja, P., and Bruce, H. Better to organize personal information by folders or by tags?: The devil is in the details. *Proceedings of the American Society for Information Science and Technology 45*, 1 (2008), 1–13.

[45] Cockburn, A., and Gutwin, C. A predictive model of human performance with scrolling and hierarchical lists. *Human-Computer Interaction 24*, 3 (2009), 273–314.

[46] Cockburn, A., Gutwin, C., and Greenberg, S. A predictive model of menu performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, ACM (New York, NY, USA, 2007), 627–636.

[47] Cockburn, A., and McKenzie, B. What do web users do? an empirical analysis of web use. *International Journal of Human-Computer Studies 54*, 6 (2001), 903–922.

[48] CocoaDev. MethodSwizzling. `http://cocoadev.com/MethodSwizzling`, retrieved 2013.

[49] Cole, I. Human aspects of office filing: Implications for the electronic office. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 26, SAGE Publications (1982), 59–63.

[50] Cook, T. It's 10 o'clock: do you know where your data are? *Technol. Rev. 98*, 1 (1995), 48–53.

[51] Cooley, R., Mobasher, B., and Srivastava, J. Web mining: information and pattern discovery on the world wide web. In *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on* (1997), 558–567.

[52] Cruz, I. F., and Xiao, H. A layered framework supporting personal information integration and application design for the semantic desktop. *The VLDB Journal 17*, 6 (2008), 1385–1406.

[53] Cutrell, E., Dumais, S., and Teevan, J. Searching to eliminate personal information management. *Communications of the ACM 49*, 1 (2006), 58–64.

[54] Cutrell, E., Robbins, D., Dumais, S., and Sarin, R. Fast, flexible filtering with phlat. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, ACM (New York, NY, USA, 2006), 261–270.

[55] de Bruijn, O., Spence, R., and Chong, M. Y. Rsvp browser: Web browsing on small screen devices. *Personal Ubiquitous Comput. 6*, 4 (2002), 245–252.

[56] Dhyani, D., Ng, W. K., and Bhowmick, S. S. A survey of web metrics. *ACM Comput. Surv. 34*, 4 (2002), 469–503.

[57] Dourish, P., Edwards, W. K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D. B., and Thornton, J. Extending document management systems with user-specific active properties. *ACM Trans. Inf. Syst. 18*, 2 (2000), 140–170.

[58] Dourish, P., Edwards, W. K., LaMarca, A., and Salisbury, M. Presto: an experimental architecture for fluid interactive document spaces. *ACM Trans. Comput.-Hum. Interact. 6*, 2 (1999), 133–161.

[59] Dropbox, Inc. Dropbox. `https://www.dropbox.com/`, retrieved 2013.

[60] Duh, H. B.-L., Tan, G. C. B., and Chen, V. H.-h. Usability evaluation for mobile device: a comparison of laboratory and field tests. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, MobileHCI '06, ACM (New York, NY, USA, 2006), 181–186.

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

[62] Dumais, S. T., and Jones, W. P. A comparison of symbolic and spatial filing. In *CHI '85: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 1985), 127–130.

[63] Ehret, B. D. Learning where to look: location learning in graphical user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, CHI '02, ACM (New York, NY, USA, 2002), 211–218.

[64] Elsweiler, D., Harvey, M., and Hacker, M. Understanding re-finding behavior in naturalistic email interaction logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in*

Information Retrieval, SIGIR '11, ACM (New York, NY, USA, 2011), 35–44.

[65] Fagin, R., Kumar, R., and Sivakumar, D. Comparing top k lists. *SIAM Journal on Discrete Mathematics 17*, 1 (2004), 134–160.

[66] Fertig, S., Freeman, E., and Gelernter, D. "Finding and reminding" reconsidered. *SIGCHI Bull. 28*, 1 (1996), 66–69.

[67] Findlater, L., and McGrenere, J. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, ACM (New York, NY, USA, 2004), 89–96.

[68] Findlater, L., Moffatt, K., McGrenere, J., and Dawson, J. Ephemeral adaptation: the use of gradual onset to improve menu selection performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 1655–1664.

[69] Fisher, D., Brush, A. J., Gleave, E., and Smith, M. A. Revisiting whittaker & sidner's "email overload" ten years later. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, CSCW '06, ACM (New York, NY, USA, 2006), 309–312.

[70] Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. 1954. *J Exp Psychol Gen 121*, 3 (September 1992), 262–269.

[71] Forlines, C., and Balakrishnan, R. Improving visual search with image segmentation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 1093–1102.

[72] Freeman, E., and Gelernter, D. Lifestreams: a storage model for personal data. *SIGMOD Rec. 25*, 1 (Mar. 1996), 80–86.

[73] Frias-Martinez, E., and Karamcheti, V. A prediction model for user access sequences. In *Proceedings of the WEBKDD workshop: web mining for usage patterns and user profiles, ACM SIGKDD international conference on knowledge discovery and data mining* (2002).

[74] Gajos, K. Z., Czerwinski, M., Tan, D. S., and Weld, D. S. Exploring the design space for adaptive graphical user interfaces. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, ACM (New York, NY, USA, 2006), 201–208.

[75] Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., and Weld, D. S. Predictability and accuracy in adaptive user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, ACM (New York, NY, USA, 2008), 1271–1274.

[76] Gaul, W., and Schmidt-thieme, L. Mining web navigation path fragments. In *In Proceedings of the Workshop on Web Mining for E-Commerce Challenges and Opportunities* (2000).

[77] Gemmell, J., Bell, G., Lueder, R., Drucker, S., and Wong, C. Mylifebits: fulfilling the memex vision. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, ACM (New York, NY, USA, 2002), 235–238.

[78] Géry, M., and Haddad, H. Evaluation of web usage mining approaches for user's next request prediction. In *Proceedings of the 5th ACM international workshop on Web information and data management*, WIDM '03, ACM (New York, NY, USA, 2003), 74–81.

[79] Gifford, D. K., Jouvelot, P., Sheldon, M. A., and O'Toole, Jr., J. W. Semantic file systems. *SIGOPS Oper. Syst. Rev. 25*, 5 (1991), 16–25.

[80] Golemati, M., Katifori, A., Giannopoulou, E. G., Daradimos, I., and Vassilakis, C. Evaluating the significance of the windows explorer visualization in personal information management browsing tasks. In *In-*

formation Visualization, 2007. IV'07. 11th International Conference, IEEE (2007), 93–100.

[81] Gonçalves, D. Users and their documents. Tech. rep., Technical Report, Instituto Superior Técnico, 2002.

[82] Gonçalves, D., and Jorge, J. An empirical study of personal document spaces. *Interactive Systems. Design, Specification, and Verification* (2003), 403–412.

[83] Google. Quick Search Box. `http://www.google.com/quicksearchbox/`, retrieved 2013.

[84] Greenberg, S. Using unix: Collected traces of 168 users. Tech. rep., Research Report 88/333/45, Department of Computer Science, University of Calgary, Calgary, Alberta, 1988.

[85] Greenberg, S., and Witten, I. Supporting command reuse: empirical foundations and principles. *International Journal of Man-Machine Studies 39*, 3 (1993), 353–390.

[86] Greenberg, S., and Witten, I. Supporting command reuse: Mechanisms for reuse. *International Journal of Man-Machine Studies 39*, 3 (1993), 391–425.

[87] Gregor, S., and Peter, K. Stability of ranked gene lists in large microarray analysis studies. *Journal of Biomedicine and Biotechnology 2010* (2010).

[88] Gustafson, S., Baudisch, P., Gutwin, C., and Irani, P. Wedge: clutter-free visualization of off-screen locations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, ACM (New York, NY, USA, 2008), 787–796.

[89] Hart, S., and Staveland, L. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Human Mental Workload*, P. Hancock, Ed. (1988), 139–183.

[90] Hearst, M. Design recommendations for hierarchical faceted search interfaces. In *ACM SIGIR workshop on faceted search* (2006), 1–5.

[91] Hearst, M. A. Clustering versus faceted categories for information exploration. *Commun. ACM 49*, 4 (2006), 59–61.

[92] Henderson, S. Genre, task, topic and time: facets of personal digital document management. In *CHINZ '05: Proceedings of the 6th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction*, ACM (New York, NY, USA, 2005), 75–82.

[93] Henderson, S. How do people manage their documents?: an empirical investigation into personal document management practices among knowledge workers. *PhD Thesis-University of Auckland* (2009).

[94] Henderson, S. Personal document management strategies. In *Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, CHINZ '09, ACM (New York, NY, USA, 2009), 69–76.

[95] Henderson, S., and Srinivasan, A. An empirical analysis of personal digital document structures. *Human Interface and the Management of Information. Designing Information Environments* (2009), 394–403.

[96] Hertzum, M. User testing in industry: A case study of laboratory, workshop, and field tests. In *Proceedings of the 5th ERCIM Workshop* (1999), 59–72.

[97] Hick, W. On the rate of gain of information. *The Quarterly Journal of Experimental Psychology 4*, 1 (1952), 11–26.

[98] Hill, W. C., Hollan, J. D., Wroblewski, D., and McCandless, T. Edit wear and read wear. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, ACM (New York, NY, USA, 1992), 3–9.

[99] Huang, Y.-F., and Hsu, J.-M. Mining web logs to improve hit ratios of prefetching and caching. *Know.-Based Syst. 21* (February 2008), 62–69.

[100] Humanized. Enso Launcher. `http://humanized.com/enso/launcher/`, retrieved 2013.

[101] Hyman, R. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology 45*, 3 (1953), 188–196.

[102] Jensen, C., Lonsdale, H., Wynn, E., Cao, J., Slater, M., and Dietterich, T. G. The life and times of files and information: a study of desktop provenance. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, ACM (New York, NY, USA, 2010), 767–776.

[103] Jones, W. Finders, keepers? the present and future perfect in support of personal information management. *First Monday 9*, 3 (2004).

[104] Jones, W. Personal information management. *Annual review of information science and technology 41*, 1 (2007), 453–504.

[105] Jones, W., Bruce, H., and Dumais, S. Keeping found things found on the web. In *Proceedings of the tenth international conference on Information and knowledge management*, ACM (2001), 119–126.

[106] Jones, W., Bruce, H., and Dumais, S. How do people get back to information on the web? how can they do it better? In *Proceedings of INTERACT '03* (2003), 793–796.

[107] Jones, W., Dumais, S., and Bruce, H. Once found, what then? A study ofkeeping behaviors in the personal use of Web information. *Proceedings of the American Society for Information Science and Technology 39*, 1 (2002), 391–402.

[108] 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: CHI '05 extended abstracts on Human factors in computing systems*, ACM (New York, NY, USA, 2005), 1505–1508.

[109] Joy, W. *An Introduction to the C shell*, vol. 2c. University of California, Berkeley, 1980.

[110] Kaasten, S., and Greenberg, S. Integrating back, history and bookmarks in web browsers. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, ACM (New York, NY, USA, 2001), 379–380.

[111] Käki, M. Findex: search result categories help users when document ranking fails. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, ACM (New York, NY, USA, 2005), 131–140.

[112] Kaptelinin, V. Creating computer-based work environments: an empirical study of macintosh users. In *Proceedings of the 1996 ACM SIGCPR/SIGMIS conference on Computer personnel research*, ACM (1996), 360–366.

[113] Karlson, A. K., Smith, G., and Lee, B. Which version is this?: improving the desktop experience within a copy-aware computing ecosystem. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 2669–2678.

[114] Koren, J., Leung, A., Zhang, Y., Maltzahn, C., Ames, S., and Miller, E. Searching and navigating petabyte-scale file systems based on facets. In *PDSW '07: Proceedings of the 2nd international workshop on Petascale data storage*, ACM (New York, NY, USA, 2007), 21–25.

[115] Krishnan, A., and Jones, S. Timespace: activity-based temporal visualisation of personal information spaces. *Personal Ubiquitous Comput. 9*, 1 (2005), 46–65.

[116] Kuiyu, B. An intelligent recommender system using sequential Web access patterns. In *Conference on Cybernetics and Intelligent Systems*, vol. 1 (2004), 3.

[117] Kurtenbach, G. P. *The design and evaluation of marking menus*. PhD thesis, University of Toronto, Ontario, Canada, 1993.

[118] Kwasnik, B. How a personal document's intended use or purpose affects its classification in an office. In *ACM SIGIR Forum*, vol. 23, ACM (1989), 207–210.

[119] Kwasnik, B. The importance of factors that are not document attributes in the organisation of personal documents. *Journal of documentation 47*, 4 (1991), 389–398.

[120] Landauer, T. K., and Nachbar, D. W. Selection from alphabetic and numeric menu trees using a touch screen: breadth, depth, and width. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '85, ACM (New York, NY, USA, 1985), 73–78.

[121] Lansdale, M. The psychology of personal information management. *Applied Ergonomics 19*, 1 (1988), 55–66.

[122] Lee, D., Choi, J., Kim, J.-H., Noh, S. H., Min, S. L., Cho, Y., and Kim, C. S. On the existence of a spectrum of policies that subsumes the least recently used (lru) and least frequently used (lfu) policies. *SIGMETRICS Perform. Eval. Rev. 27*, 1 (1999), 134–143.

[123] Lewis, J. P., Rosenholtz, R., Fong, N., and Neumann, U. Visualids: automatic distinctive icons for desktop interfaces. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, ACM (New York, NY, USA, 2004), 416–423.

[124] Mackay, W. E. Diversity in the use of electronic mail: a preliminary inquiry. *ACM Trans. Inf. Syst. 6*, 4 (Oct. 1988), 380–397.

[125] Mackay, W. E. The missing link: integrating paper and electronic documents. In *Proceedings of the 15th French-speaking conference on human-computer interaction*, IHM 2003, ACM (New York, NY, USA, 2003), 1–8.

[126] Malacria, S., Scarr, J., Cockburn, A., Gutwin, C., and Grossman, T. Skillometers: reflective widgets that motivate and help users to improve performance. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, UIST '13, ACM (New York, NY, USA, 2013), 321–330.

[127] Malone, T. W. How do people organize their desks?: Implications for the design of office information systems. *ACM Trans. Inf. Syst. 1*, 1 (1983), 99–112.

[128] Mander, R., Salomon, G., and Wong, Y. Y. A 'pile' metaphor for supporting casual organization of information. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 1992), 627–634.

[129] Mannila, H., and Toivonen, H. Discovering generalized episodes using minimal occurrences. In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining* (1996), 146–151.

[130] Marchionini, G. Toward human-computer information retrieval. *Bulletin of the American Society for Information Science and Technology 32*, 5 (2006), 20–22.

[131] Markatos, E. P. On caching search engine query results. *Computer Communications 24*, 2 (2001), 137–143.

[132] Markov, A. The theory of algorithms. *American Mathematical Society Translations* (1960).

[133] Marsden, G., and Cairns, D. E. Improving the usability of the hierarchical file system. In *SAICSIT '03: Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, South African Institute for Computer Scientists and Information Technologists (Republic of South Africa, 2003), 122–129.

[134] Matejka, J., Li, W., Grossman, T., and Fitzmaurice, G. Communitycommands: command recommendations for software applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ACM (2009), 193–202.

[135] Mayer, M. Web history tools and revisitation support: A survey of existing approaches and directions. *Found. Trends Hum.-Comput. Interact. 2*, 3 (Mar. 2009), 173–278.

[136] McKenzie, B., and Cockburn, A. An Empirical Analysis of Web Page Revisitation. In *Maui, Hawaii: Proceedings of the 34th Hawaiian International Conference on System Sciences, HICCSS34* (2001).

[137] Megiddo, N., and Modha, D. S. Arc: A self-tuning, low overhead replacement cache. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, USENIX Association (Berkeley, CA, USA, 2003), 115–130.

[138] Megiddo, N., and Modha, D. S. Outperforming lru with an adaptive replacement cache algorithm. *Computer 37* (2004), 58–65.

[139] Microsoft. Windows Search. http://windows.microsoft.com/en-US/windows7/products/features/windows-search, retrieved 2013.

[140] Miller, D. P. The depth/breadth tradeoff in hierarchical computer menus. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 25*, 1 (1981), 296–300.

[141] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on* (2002), 669–672.

[142] Moon, J. M., and Fu, W.-T. Effects of spatial locations and luminance on finding and re-finding information in a desktop environment. In *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, ACM (New York, NY, USA, 2009), 3365–3370.

[143] Mozilla. The Places frecency algorithm. `https://developer.mozilla.org/en/The_Places_frecency_algorithm`, 2008.

[144] Mozilla. Firefox web browser. `http://www.mozilla.com/firefox/`, retrieved 2013.

[145] Nardi, B., Anderson, K., and Erickson, T. Filing and finding computer files. *Proceedings of the East-West HCI, Moscow, Russia* (1995).

[146] Nielsen, C. M., Overgaard, M., Pedersen, M. B., Stage, J., and Stenild, S. It's worth the hassle!: the added value of evaluating the usability of mobile systems in the field. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*, NordiCHI '06, ACM (New York, NY, USA, 2006), 272–280.

[147] Obendorf, H., Weinreich, H., Herder, E., and Mayer, M. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 2007), 597–606.

[148] Objective Development. LaunchBar. `http://www.obdev.at/products/launchbar/`, retrieved 2013.

[149] Padioleau, Y., Sigonneau, B., and Ridoux, O. Lisfs: a logical information system as a file system. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, ACM (New York, NY, USA, 2006), 803–806.

[150] Perkowitz, M., and Etzioni, O. Towards adaptive web sites: Conceptual framework and case study. *Computer Networks 31*, 11-16 (1999), 1245–1258.

[151] Perugini, S. Supporting multiple paths to objects in information hierarchies: Faceted classification, faceted search, and symbolic links. *Inf. Process. Manage. 46*, 1 (2010), 22–43.

[152] Philippe Mougin. F-Script. `http://www.fscript.org`, retrieved 2013.

[153] Quan, D., Bakshi, K., Huynh, D., and Karger, D. User interfaces for supporting multiple categorization. In *INTERACT'03; IFIP TC13 International Conference on Human-Computer Interaction, 1st-5th September 2003, Zurich, Switzerland*, Ios Pr Inc (2003), 228.

[154] Raskin, J. *The humane interface: new directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000.

[155] Ravasio, P., Schär, S. G., and Krueger, H. In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Trans. Comput.-Hum. Interact. 11*, 2 (2004), 156–180.

[156] Rekimoto, J. Time-machine computing: a time-centric approach for the information environment. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, ACM (New York, NY, USA, 1999), 45–54.

[157] Ringel, M., Cutrell, E., Dumais, S., and Horvitz, E. Milestones in time: The value of landmarks in retrieving information from personal stores. In *Human-computer interaction: INTERACT'03; IFIP TC13 International Conference on Human-Computer Interaction, 1st-5th September 2003, Zurich, Switzerland*, Ios Pr Inc (2003), 184.

[158] Robertson, G. G., Mackinlay, J. D., and Card, S. K. Cone trees: animated 3d visualizations of hierarchical information. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 1991), 189–194.

[159] Robinson, J. T., and Devarakonda, M. V. Data cache management using frequency-based replacement. *SIGMETRICS Perform. Eval. Rev. 18*, 1 (1990), 134–142.

[160] Romano, S., and ElAarag, H. A quantitative study of recency and frequency based web cache replacement strategies. In *CNS '08: Proceedings of the 11th communications and networking simulation symposium*, ACM (New York, NY, USA, 2008), 70–78.

[161] Running with Crayons Ltd. Alfred. `http://www.alfredapp.com/`, retrieved 2013.

[162] Russell, D., and Lawrence, S. Search everything. *Personal information management* (2007), 153–166.

[163] Salton, G., and McGill, M. J. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[164] Sarukkai, R. Link prediction and path analysis using Markov chains. *Computer Networks 33*, 1-6 (2000), 377–386.

[165] Scarr, J., Cockburn, A., Gutwin, C., and Quinn, P. Dips and ceilings: understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors*

in Computing Systems, CHI '11, ACM (New York, NY, USA, 2011), 2741–2750.

[166] Sears, A., and Shneiderman, B. Split menus: effectively using selection frequency to organize menus. *ACM TOCHI 1*, 1 (1994), 27–51.

[167] Sinha, D., and Basu, A. Gardener: A file browser assistant to help users maintaining semantic folder hierarchy. In *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*, IEEE (2012), 1–6.

[168] Solomon, M. SIMBL. `http://www.culater.net/software/SIMBL/SIMBL.php`, retrieved 2013.

[169] Soules, C. A. N., and Ganger, G. R. Connections: using context to enhance file search. *SIGOPS Oper. Syst. Rev. 39*, 5 (2005), 119–132.

[170] Stasko, J. An evaluation of space-filling information visualizations for depicting hierarchical structures. *Int. J. Hum.-Comput. Stud. 53*, 5 (Nov. 2000), 663–694.

[171] Steve Nygard. class-dump. `http://www.codethecode.com/projects/class-dump/`, retrieved 2013.

[172] Tak, S. *Understanding and Supporting Window Switching*. PhD thesis, University of Canterbury, 2011.

[173] Tang, J. C., Drews, C., Smith, M., Wu, F., Sue, A., and Lau, T. Exploring patterns of social commonality among file directories at work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, ACM (New York, NY, USA, 2007), 951–960.

[174] Tauscher, L., and Greenberg, S. How people revisit web pages: empirical findings and implications for the design of history systems. *International Journal of Human Computer Studies 47* (1997), 97–138.

[175] Teevan, J., Adar, E., Jones, R., and Potts, M. A. S. Information re-retrieval: repeat queries in yahoo's logs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, ACM (New York, NY, USA, 2007), 151–158.

[176] 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 *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (New York, NY, USA, 2004), 415–422.

[177] Thorington, J., and Irwin, J. An adaptive replacement algorithm for paged-memory computer systems. *IEEE Transactions on Computers 21* (1972), 1053–1061.

[178] Treisman, A. Perceptual grouping and attention in visual search for features and for objects. *Journal of Experimental Psychology: Human Perception and Performance 8*, 2 (1982), 194.

[179] Tulving, E., and Thomson, D. Encoding specificity and retrieval processes in episodic memory. *Psychological review 80*, 5 (1973), 352–373.

[180] Tyler, S. K., and Teevan, J. Large scale query log analysis of re-finding. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, ACM (New York, NY, USA, 2010), 191–200.

[181] Webber, W., Moffat, A., and Zobel, J. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS) 28*, 4 (2010), 20.

[182] Weinreich, H., Obendorf, H., Herder, E., and Mayer, M. Not quite the average: An empirical study of web use. *ACM Trans. Web 2*, 1 (2008), 1–31.

[183] Whittaker, S., and Hirschberg, J. The character, value, and management of personal paper archives. *ACM Trans. Comput.-Hum. Interact. 8*, 2 (2001), 150–170.

[184] Whittaker, S., Matthews, T., Cerruti, J., Badenes, H., and Tang, J. Am i wasting my time organizing email?: a study of email refinding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 3449–3458.

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

[186] Wu, S., and Crestani, F. Methods for ranking information retrieval systems without relevance judgments. In *Proceedings of the 2003 ACM symposium on Applied computing*, SAC '03, ACM (New York, NY, USA, 2003), 811–816.

[187] Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM (New York, NY, USA, 2003), 401–408.

[188] Zhang, J., Izmailov, R., Reininger, D., and Ott, M. Web caching framework: Analytical models and beyond. In *Internet Applications, 1999. IEEE Workshop on*, IEEE (2002), 132–141.

[189] Zipf, G. K. *Human Behaviour and the Principle of Least-Effort.* Addison-Wesley, 1949.

Appendices

Appendix A

Characterisation Study Material

The following material from the file retrieval characterisation study in Chapter 5 is presented on the following pages:

1. The information sheet provided to participants before they agreed to participate in the study.

2. The consent form that was signed by all participants before FileMonitor was installed on their computers.

3. The template for the post-study interview. This template was used by the interviewer, both as a topic prompt and to take notes during the interview. However, it was only used as a guide, and additional questions were generally asked based on participant responses to the planned questions.

Department of Computer Science and Software Engineering

Stephen Fitchett
Erskine Room 344, Tel: +64 27 3788035
Email: stephen.fitchett@pg.canterbury.ac.nz

Professor Andy Cockburn
Tel: +64 3 364 2987 x7755
Email: andy@cosc.canterbury.ac.nz

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

Information Sheet for "File Use Study"

You are invited to participate in the research project "File Use Study".

The aim of this project is to understand how users access their files. This will allow us to design and build improved user interfaces for file access.

If you agree to participate in the study you will be asked to install our file logging software on your computer. This program silently monitors the mechanisms for file accesses (e.g., by opening folders, by double-clicking on a file icon, use of search, file renaming activities, etc.) and records them in log files stored on your computer.

Privacy

Information that includes file names, file paths and Spotlight search terms are stored, which allows us to see how frequently the same files are used, but the actual file content is *not* stored. Your log files will be processed by a computer program, and no human will see your file names or search terms. You can view the content of the log files at any point by viewing the content of the files in "~/Library/Application Support/FileLogger/". At any time, we will be happy to walk you through the content of these files. You also have the right to withdraw from the project at any time, including withdrawal of any information provided.

What's involved

At the start of the study we'll install the software, and you'll complete a short pre-study survey. The software will gather data for a period of four weeks. This should have no effect on your computer's performance, and you should continue to use your computer as normal.

At the end of the study we will contact you to arrange a time to retrieve the log files and uninstall the logging software. There will be a short post-study questionnaire. We offer a $50 shopping voucher as reward and thanks to all participants when we receive their log files.

About the study

The results of the project may be published (including in a publicly accessible PhD thesis), but you are assured of the complete confidentiality of data gathered in this investigation: the identity of participants will not be made public. To ensure anonymity and confidentiality your signed consent form will be stored in a locked filing cabinet in a locked office, and computer logs of your participation are anonymous.

The project is being carried out by PhD student Stephen Fitchett, as part of a Marsden funded research project under the supervision of Professor Andy Cockburn. Both Stephen and Andy can be

contacted using the information above. They will be pleased to discuss any concerns you may have about participation in the project.

The project has been reviewed and approved by the University of Canterbury Human Ethics Committee.

Technical issues

If you notice any problems as a result of the sofware (such as Finder restarting itself or Finder features not working correctly), please contact Stephen. In the unlikely event that you have major issues that you need to resolve immediately, you can temporarily disable the logging tool by entering the following three commands exactly in a Terminal window (you can access the Terminal by, for example, searching for it in Spotlight or navigating to /Applications/Utilities/Terminal):

```
cd ~/Library/Application\ Support/SIMBL/Plugins
mv FileLogger.bundle ../FileLogger.bundle
killall Finder
```

Please contact Stephen as soon as possible if you have had to do this.

Yours sincerely

Stephen Fitchett
PhD Student

Andy Cockburn
Professor

Department of Computer Science and Software Engineering

Stephen Fitchett
Erskine Room 344, Tel: +64 27 3788035
Email: stephen.fitchett@pg.canterbury.ac.nz

Professor Andy Cockburn
Tel: +64 3 364 2987 x7755
Email: andy@cosc.canterbury.ac.nz

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

29th March, 2012

Consent form for "File Use Study"

I have read and understood the description of the above-named project. On this basis I agree to participate as a subject in the project, and I consent to publication of the results of the project with the understanding that anonymity will be preserved. I am over 18 years of age.

I understand also that I may at any time withdraw from the project, including withdrawal of any information I have provided.

I note that the project has been reviewed and approved by the University of Canterbury Human Ethics Committee.

NAME (please print): _____

Signature: _____

Date: _____

Age: _____

Gender: ☐ Male ☐ Female

Which best describes you?

☐ CSSE undergrad ☐ CSSE postgrad ☐ CSSE staff ☐ Other
☐ Other undergrad ☐ Other postgrad ☐ Other UC staff

How often do you use your computer on a typical day?

☐ Less than one hour ☐ 1-2 hours ☐ 2-4 hours ☐ 4-8 hours ☐ 8 hours+

File use study survey **Participant:** _____

Introduction: We're interested in the different ways people find their files. We've prepared some questions, but feel free to add any other information you think might be relevant. If there are any questions you're uncomfortable with, just say so and we can continue.

1. Would you be able to identify the three files you've been working with most over the last week? (is this indicative of your normal usage? If not, how do you usually go about opening files?)
   (for each:
   - See how they get to it.
   - If it's open:
     - ask them how long it has been open for
     - ask when they close the file and why they do this
     - ask them to close it and show you how they would open it again
   - If it's closed:
     - ask them to show you how they open it.)

   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____

2. Can you think of a file you've recently opened that was hard to find? Can you show me how you'd go about finding and opening it?

   _____
   _____
   _____
   _____

   - (if they used search) Now, can you show me how you'd go about finding it if search wasn't available?

     _____
     _____
     _____
     _____
     _____

3. OK, so you <list>. Are there any other features you use to find or open files?

4. (If multiple techniques mentioned) What, if anything, determines what tool you use to find or open files?

5. (If both search and other methods mentioned) In what situations would you search, rather than navigate to a file or application?

Search*:*

6. (if search not yet mentioned) Do you ever search for files?

7. (if yes or previously mentioned) What search interfaces do you use to find files? OR (if some already mentioned) Apart from <list already mentioned>, are they any other search interfaces you use to find files?

8. (for omitted search techniques) Do you ever [use Spotlight/search for files in Finder]?

Other techniques:

9. Do you use the Finder sidebar, and if so, how? Have you customised it?

10. How do you use the desktop? (in particular, is it used for convenient access to things, or unorganised temporary storage, or something else? Roughly how much is on it?)

    _____

    _____

    _____

11. (if recent items not mentioned) Do you ever use any menus or interfaces that show recent items? (if yes) Which ones?

    _____

    _____

    _____

    - (for omitted techniques) Do you ever use [the recent items menu in the Apple menu/the open recent menu in applications/other open recent interfaces]?

      _____

      _____

12. (if open dialog not mentioned) Do you ever use the open dialog in applications? (if yes) Under what circumstances would you use it?

    _____

    _____

13. (if dock not mentioned) Do you use the Dock to open applications or other items? (if yes) How would you normally use it in day to day use? Have you customised the Dock?

    _____

    _____

14. (if command line not mentioned) Do you ever use the command line to open files? (if yes) Under what circumstances would you use it?

    _____

    _____

15. (if third party tools not mentioned) Do you ever use third party tools to open files? (if yes) Which ones? Under what circumstances would you use them?

    _____

    _____

    - (if yes) Can you think of a file you've recently opened with this tool? Can you show me how you'd use it to find the file?

      _____

      _____

      _____

16. What annoyances, if any, do you have when finding or opening files?

    _____

    _____

    _____

    - (if not referred to) when using Finder?

      _____

      _____

      _____

- (if not referred to) when using search?

  _____

  _____

  _____

17. Can you think of any changes you'd like made to the interfaces you use to find files?

    _____

    _____

    _____

    _____

Appendix B

Improved File Retrieval Interfaces – Lab Study Material

The following material from the lab study of Icon Highlights, Search Directed Navigation and Hover Menus in Chapter 7 is presented on the following pages:

1. The consent form that was signed by all participants before participating in the study.

2. The pre-experiment and post-experiment surveys.

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF SASKATCHEWAN
INFORMED CONSENT FORM

Research Project: **Using Navigation Aids to Help Locate Files**

Investigators: Dr. Carl Gutwin, Department of Computer Science
Stephen Fitchett, Department of Computer Science
Roxanne Dowd, Department of Computer Science

This consent form, a copy of which has been given to you, is only part of the process of informed consent. It should give you the basic idea of what the research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, please ask. Please take the time to read this form carefully and to understand any accompanying information.

The goal of the research is to understand the effects of different navigation methods on locating files.

The session will require approximately an hour, during which you will be asked to locate files in a mock file browser.

At the end of the session, you will be given more information about the purpose and goals of the study, and there will be time for you to ask questions about the research.

The data collected from this study will be used in articles for publication in journals and conference proceedings.

As one way of thanking you for your time, we will be pleased to make available to you a summary of the results of this study once they have been compiled (usually within three months). This summary will outline the research and discuss our findings and recommendations. If you would like to receive a copy of this summary, please write down your email address here.

Contact email address:_____

All personal and identifying data will be kept confidential. If explicit consent has been given, textual excerpts, photographs, or video recordings may be used in the dissemination of research results in scholarly journals or at scholarly conferences. Anonymity will be preserved by using pseudonyms in any presentation of textual data in journals or at conferences. The informed consent form and all research data will be kept in a secure location under confidentiality in accordance with University policy for 5 years post publication. Do you have any questions about this aspect of the study?

You are free to withdraw from the study at any time without penalty and without losing any advertised benefits. Withdrawal from the study will not affect your academic status or your access to services at the university. If you withdraw, your data will be deleted from the study and destroyed.

Your continued participation should be as informed as your initial consent, so you should feel free to ask for clarification or new information throughout your participation. If you have further questions concerning matters related to this research, please contact:

- Dr. Carl Gutwin, Professor, Dept. of Computer Science, (306) 966-8646, gutwin@cs.usask.ca

Your signature on this form indicates that you have understood to your satisfaction the information regarding participation in the research project and agree to participate as a participant. In no way does this waive your legal rights nor release the investigators, sponsors, or involved institutions from their legal and professional responsibilities. If you have further questions about this study or your rights as a participant, please contact:

- Dr. Carl Gutwin, Professor, Dept. of Computer Science, (306) 966-8646, gutwin@cs.usask.ca
- Office of Research Services, University of Saskatchewan, (306) 966-4053

Participant's signature:_____

Date:_____

Investigator's signature:_____

Date:_____

A copy of this consent form has been given to you to keep for your records and reference. This research has the ethical approval of the Office of Research Services at the University of Saskatchewan.

Participant number: _____

Pre-Experiment Survey

Gender: ☐ Male ☐ Female

Age: _____

Do you have normal/corrected to normal vision?

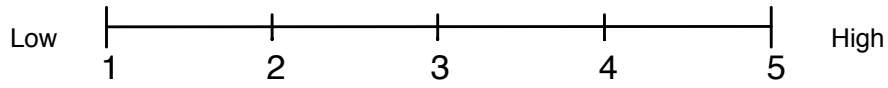
☐ Yes ☐ No

How many hours do you use computers on a typical day? _____

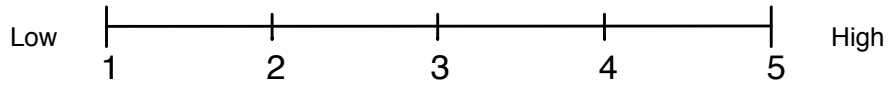
Participant number: _____

Standard icon view

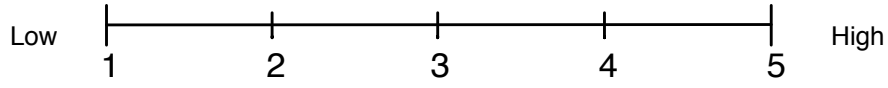
Mental demand *How mentally demanding was the task?*



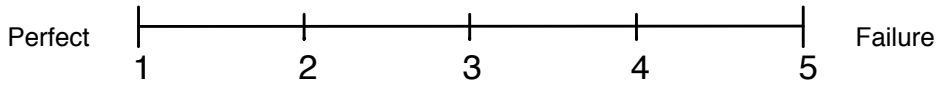
Physical demand *How physically demanding was the task?*



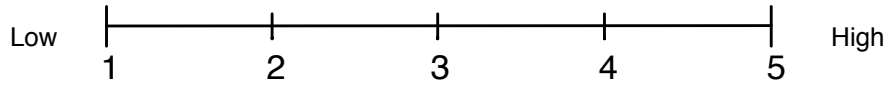
Temporal demand *How hurried or rushed was the pace of the task?*



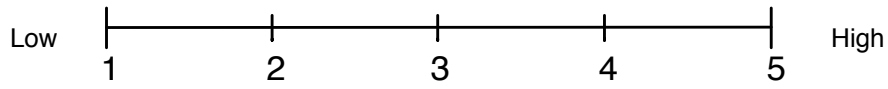
Performance *How successful were you in accomplishing what you were asked to do?*



Effort *How hard did you have to work to accomplish your level of performance?*



Frustration *How insecure, discouraged, irritated, stressed and annoyed were you?*



Comments: _____

_____

_____

_____

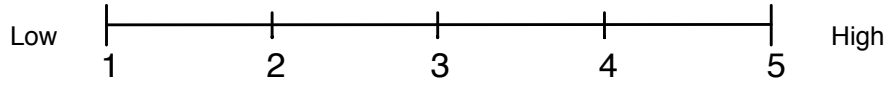
_____

_____

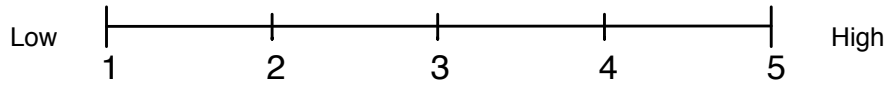
Participant number: _____

Item highlights

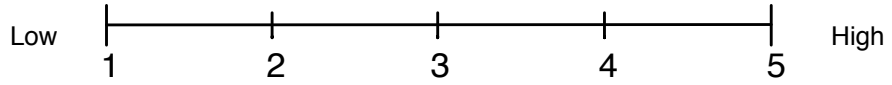
Mental demand — *How mentally demanding was the task?*



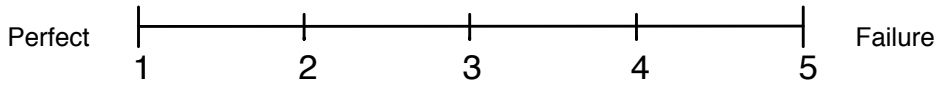
Physical demand — *How physically demanding was the task?*



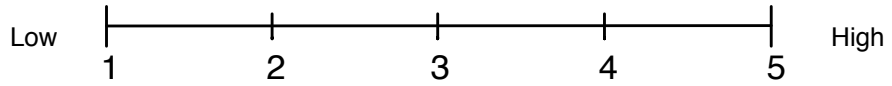
Temporal demand — *How hurried or rushed was the pace of the task?*



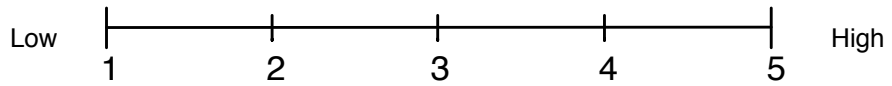
Performance — *How successful were you in accomplishing what you were asked to do?*



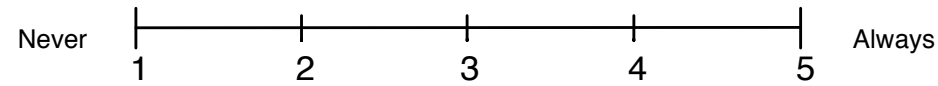
Effort — *How hard did you have to work to accomplish your level of performance?*



Frustration — *How insecure, discouraged, irritated, stressed and annoyed were you?*



I used the features provided by this navigational aid:



Comments: _____

_____

_____

_____

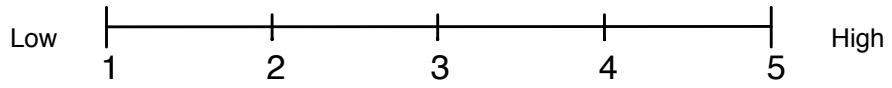
_____

_____

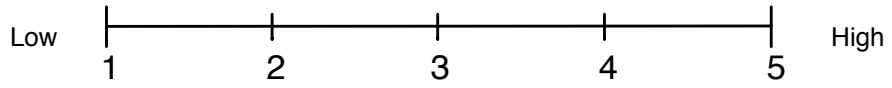
Participant number: _____

Search directed navigation

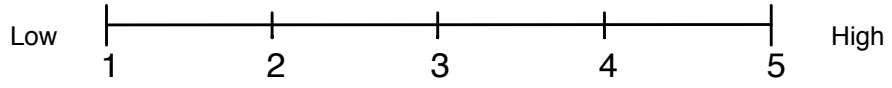
Mental demand *How mentally demanding was the task?*



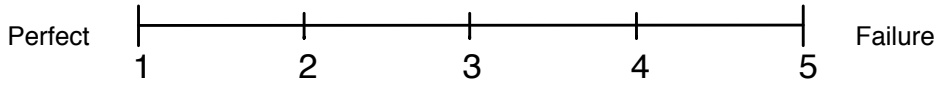
Physical demand *How physically demanding was the task?*



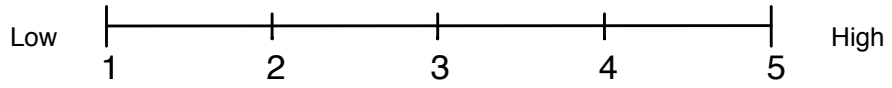
Temporal demand *How hurried or rushed was the pace of the task?*



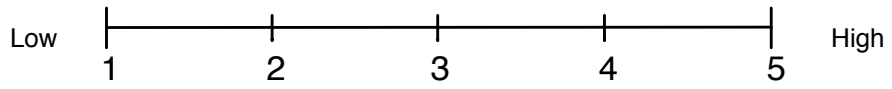
Performance *How successful were you in accomplishing what you were asked to do?*



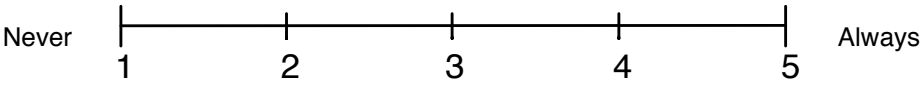
Effort *How hard did you have to work to accomplish your level of performance?*



Frustration *How insecure, discouraged, irritated, stressed and annoyed were you?*



I used the features provided by this navigational aid:



Comments: _____

_____

_____

_____

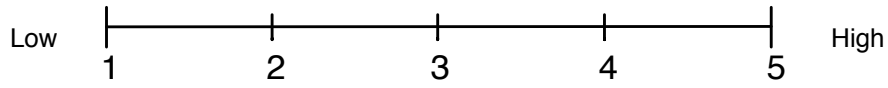
_____

_____

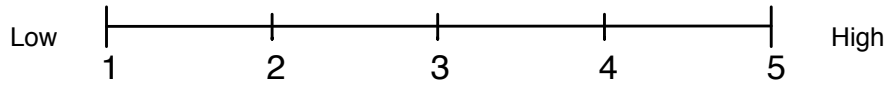
Participant number: _____

Hover menus

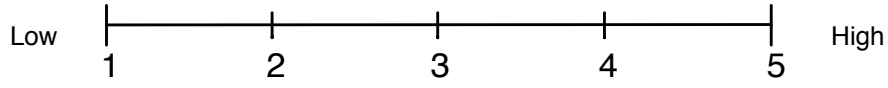
Mental demand — *How mentally demanding was the task?*



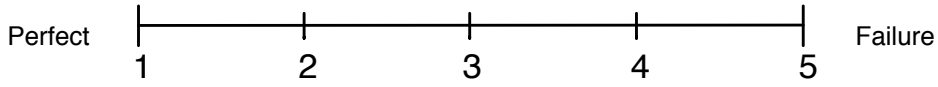
Physical demand — *How physically demanding was the task?*



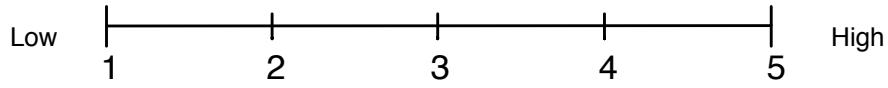
Temporal demand — *How hurried or rushed was the pace of the task?*



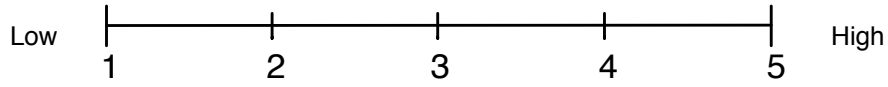
Performance — *How successful were you in accomplishing what you were asked to do?*



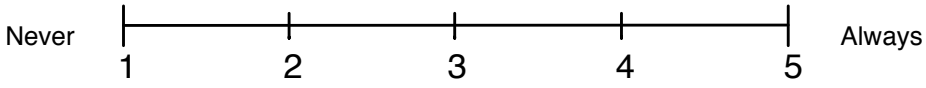
Effort — *How hard did you have to work to accomplish your level of performance?*



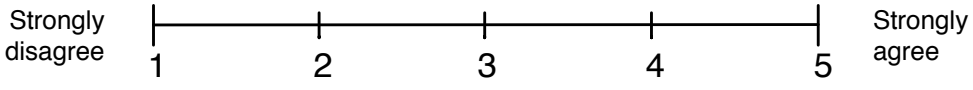
Frustration — *How insecure, discouraged, irritated, stressed and annoyed were you?*



I used the features provided by this navigational aid:



I sometimes forgot about this feature:



Comments: _____

_____

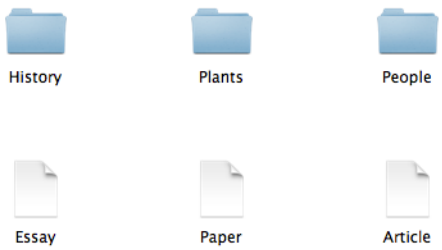
_____

_____

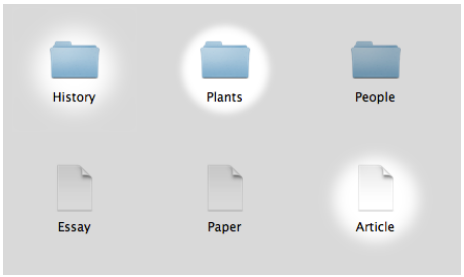
_____

Participant number: _____

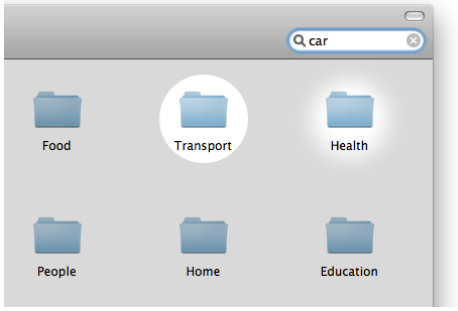
Please rank the interfaces in order from 1 to 4 (with 1 being the best) for each of the following.



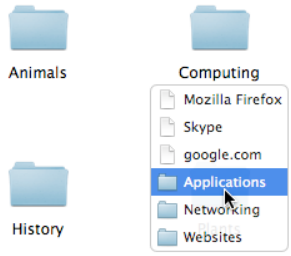
Standard icon view



Item highlights



Search Directed Navigation



Hover menus

1. Which interface was fastest to find files? (rank 1 to 4)

☐ Standard icon view ☐ Item highlights
☐ Search Directed Navigation ☐ Hover menus

2. Which interface did you make the fewest errors with? (rank 1 to 4)

☐ Standard icon view ☐ Item highlights
☐ Search Directed Navigation ☐ Hover menus

3. Which interface did you prefer overall? (rank 1 to 4)

☐ Standard icon view ☐ Item highlights
☐ Search Directed Navigation ☐ Hover menus

If I could choose between Search Directed Navigation, and a search feature like those in modern computer systems, I would prefer:

☐ Search Directed Navigation ☐ Search

Comments: _____

_____

_____

_____

Appendix C

Improved File Retrieval Interfaces – Field Study Material

The following material from the Finder Highlights field study in Chapter 8 is presented on the following pages:

1. The information web page and electronic consent form, visible to all potential participants.

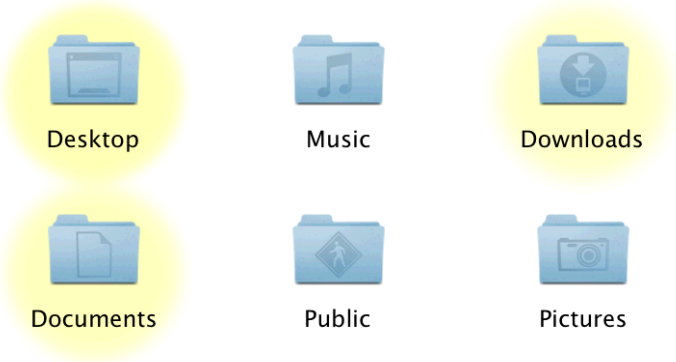
2. The pre-study survey, completed before installation of Finder Highlights.

3. The post-study survey, completed after uninstallation of Finder Highlights.

Finder Highlights Study

You are invited to participate in the research project "Finder Highlights Study".

The aim of this project is to evaluate new file retrieval interfaces in a real world field study, in order to understand whether they are effective and how they are used.



If you agree to participate in the study you will be asked to install our software on your computer for about a month. This software consists of a plugin to Finder, the file browser application on Mac OS X. It will add additional features that you can use during the study period, that are intended to aid file retrieval. These are demonstrated in an included tutorial video. It also monitors how you use Finder to navigate and retrieve files and records these details anonymously in log files stored on your computer (described below).

Eligibility

To participate in this study, you must have a Mac running Mac OS X 10.8 (Mountain Lion) that you will use for the next month. Ideally this should be your primary computer. If you have multiple Macs, please only install it on the one you use the most. You should not participate if other people use your computer under the same user account (if they use other accounts, that's ok), or if you use any other Finder plugin that modifies its appearance or behaviour.

You should be fluent in English, although it does not need to be your native language.

Privacy

As part of the study, the software will log information about how you use Finder. However, this information does not include any personal or identifying information such as filenames or search text. If you wish, you can view the content of the log files at any point by viewing the content of the files in *~/Library/Application Support/FinderHighlights Logs/*. At the end of the study, we will provide instructions for you to send us these log files; they will never be transmitted automatically. You have the right to withdraw from the study at any time, including withdrawal of any information provided.

What's involved

At the start of the study, you'll complete a short pre-study survey, download and install the software, and watch a short tutorial video explaining the new features that it provides.

For the following four weeks, you will continue to use your computer as normal, but with our software installed. You can use the features offered by our software as much or as little as you like.

At the end of this period, we will contact you with instructions on how to uninstall the software and send us your log files. There will also be a short online post-study survey.

About the study

The results of the project may be published (including in a publicly accessible PhD thesis), but you are assured of the complete confidentiality of data gathered in this investigation: the identity of participants will not be made public. To ensure anonymity and confidentiality your information will

be held securely in electronic form, and computer logs of your participation are anonymous.

The project is being carried out by PhD student Stephen Fitchett, as part of a Marsden funded research project under the supervision of Professor Andy Cockburn. Both Stephen and Andy can be contacted using the information below. They will be pleased to discuss any concerns you may have about participation in the project.

Email (Stephen): stephen.fitchett@pg.canterbury.ac.nz
Email (Andy): andy@cosc.canterbury.ac.nz
Erskine Room 344, University of Canterbury, New Zealand
Tel: +64 3 364 2987 x7755

The project has been reviewed and approved by the University of Canterbury Human Ethics Committee.

Technical issues

If you notice any problems as a result of the sofware (such as Finder restarting itself or Finder features not working correctly), please contact Stephen. In the unlikely event that you have major issues that you need to resolve immediately, you can temporarily disable the logging tool by entering the following three commands exactly in a Terminal window (you can access the Terminal by, for example, searching for it in Spotlight or navigating to /Applications/Utilities/Terminal):

```
cd ~/Library/Application\ Support/SIMBL/Plugins
mv FinderHighlights.bundle ../FinderHighlights.bundle
killall Finder
```

Please contact Stephen as soon as possible if you have had to do this.

Consent Form

I have read and understood the description of this project. On this basis I agree to participate as a subject in the project, and I consent to publication of the results of the project with the understanding that anonymity will be preserved. I am over 18 years of age.

I understand also that I may at any time withdraw from the project, including withdrawal of any information I have provided.

I note that the project has been reviewed and approved by the University of Canterbury Human Ethics Committee.

Setup and installation

To setup the software and begin the study, follow the steps below as they appear.

☐ I have read the study information and consent form and give my consent to participate in this study.

Finder Highlights: Pre-Study Survey

Make sure to read the study information and consent form before completing this survey (available at http://cosc.canterbury.ac.nz/highlights). Please be ready to download and install the Finder Highlights software when you submit this form.
* Required

Consent *

☐ I have read the study information and consent form and give my consent to participate in this study.

Contact Information

Name *
Your identity will be kept confidential and will only be known to those involved in the study. It will not be published.

Email *
Your email address will only be used for correspondence related to the study.

Demographic Information

The following information is used for statistical information only.

Age *

Gender *

○ Male
○ Female
○ Other/refuse

Roughly how many hours do you use your computer on a typical day? *
Answer for the computer you will be using for the study.

What type of computer will you be using for the study? *

○ Desktop
○ Laptop

Are you: *
Choose the option that best describes you.

○ An undergraduate student
○ A graduate or postgraduate student

- ○ University staff or researcher (including post-doc)
- ○ Employed outside a university
- ○ Other

IT experience

- ☐ I study or work in an IT related field (computer science, software development, etc)

File Organisation

How organised do you normally keep your files? *

- ○ Very disorganised
- ○ Slightly disorganised
- ○ Neither organised nor disorganised
- ○ Slightly organised
- ○ Very organised

Typically, how easy or hard is it for you to locate your files? *

- ○ Very difficult
- ○ Slightly difficult
- ○ Neither easy nor difficult
- ○ Slightly easy
- ○ Very easy

Which of the following most accurately describes how you store files? *

- ○ I keep all my files in a well organised structure
- ○ I organise many of my files, but many others are unfiled
- ○ I don't organise my files much, and most of my files are unfiled

How often do you use the following methods to retrieve files on your computer? *

	Never	Occasionally	Sometimes	Often	Very often	Not sure
Navigate through the file hierarchy in Finder	○	○	○	○	○	○
Search	○	○	○	○	○	○
"Open Recent" menus or similar	○	○	○	○	○	○
"Open" dialogs (i.e. File > Open)	○	○	○	○	○	○

Comments

If you have any comments on any of these questions, please write them here.

Submit

Never submit passwords through Google Forms.

Powered by Google Drive

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms

Finder Highlights: Post-Study Survey

Thank you for your participation in the Finder Highlights study. This survey will ask questions about your experience using the software. Your answers will be kept confidential. The survey will take approximately 10 minutes.

When completing the survey, think about how you used Icon Highlights and Search Directed Navigation when you retrieved files, compared to how you normally would have retrieved files without these features.

Recall that Icon Highlights are the yellow highlights that appear automatically for items you've previously accessed, and Search Directed Navigation is the search mode that highlights items that match a filename query.

* Required

Name *
Your identity will be kept confidential and will only be known to those involved in running the study. It will not be published.

Icon Highlights

I found Icon Highlights (the yellow highlights) useful *

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

The presence of Icon Highlights meant I didn't have to think as much when navigating to files *

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

I would like a feature similar to Icon Highlights to be a permanent feature of my file browser *

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

Compared to normal, did Icon Highlights affect whether you made errors navigating to files? *
An example of an error might be opening a folder that doesn't contain the item you are looking for.

- ○ I made fewer errors because of Icon Highlights

- ◯ Icon Highlights didn't affect whether I made errors
- ◯ I made more errors because of Icon Highlights
- ◯ Don't know

Comments on above questions

When did you find Icon Highlights most useful (if you found them useful)?

Were there any aspects of Icon Highlights that you disliked or that could be improved?

Search Directed Navigation

I found Search Directed Navigation useful *

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree

The availability of Search Directed Navigation affected how I retrieved files *

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree

I would like a feature similar to Search Directed Navigation to be a permanent feature of my file browser *

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree

Comments on above questions

Which of the following best describes how you think of Search Directed Navigation? *
Note: "Navigation" refers to navigating through the file hierarchy

- ○ A search feature
- ○ Mostly a search feature, but partly a navigation feature
- ○ Equally a search and navigation feature
- ○ Mostly a navigation feature, but partly a search feature
- ○ A navigation feature
- ○ Don't know

If you didn't use Search Directed Navigation much, why not? (tick all applicable)

- ☐ I already knew where my files were
- ☐ I couldn't remember the filenames of what I was looking for
- ☐ I forgot about it or didn't think to use it
- ☐ I don't use search much
- ☐ I don't navigate to files much
- ☐ It was too slow to find results
- ☐ I prefer using the mouse than typing a query
- ☐ I didn't understand how it worked
- ☐ Not applicable
- ☐ Don't know
- ☐ Other:

Comments on above questions

When did you find Search Directed navigation most useful (if you found it useful)?

Were there any aspects of Search Directed Navigation that you disliked or that could be improved?

Performance

The presence of Icon Highlights meant that overall I retrieved files: *

- More quickly
- At about the same speed
- More slowly
- Don't know/not applicable

The presence of Icon Highlights meant that I retrieved previously visited files: *

- More quickly
- At about the same speed
- More slowly
- Don't know/not applicable

The presence of Icon Highlights meant that I retrieved rarely visited or new files: *

- More quickly
- At about the same speed
- More slowly
- Don't know/not applicable

The presence of Search Directed Navigation meant that overall I retrieved files: *

- ◯ More quickly
- ◯ At about the same speed
- ◯ More slowly
- ◯ Don't know/not applicable

The presence of Search Directed Navigation meant that I retrieved previously visited files: *

- ◯ More quickly
- ◯ At about the same speed
- ◯ More slowly
- ◯ Don't know/not applicable

The presence of Search Directed Navigation meant that I retrieved rarely visited or new files: *

- ◯ More quickly
- ◯ At about the same speed
- ◯ More slowly
- ◯ Don't know/not applicable

Comments

Use of Different Methods

I navigated while paying attention to Icon Highlights when I otherwise would have (tick all applicable):
"Navigation" refers to navigating through the file hierarchy in Finder

- ☐ Used navigation
- ☐ Used search
- ☐ Use an "Open Recent" menu or similar
- ☐ Used an "Open" dialog
- ☐ I didn't pay attention to Icon Highlights
- ☐ Don't know
- ☐ Other:

I used Search Directed Navigation when I otherwise would have (tick all applicable):

- ☐ Used navigation
- ☐ Used search
- ☐ Use an "Open Recent" menu or similar

- [ ] Used an "Open" dialog
- [ ] I didn't use Search Directed Navigation
- [ ] Don't know
- [ ] Other:

Comments

My use of search changed as follows because of the availability of Icon Highlights: *
Search includes the Spotlight menu, the standard Finder search, or any other file search tools, EXCLUDING Search Directed Navigation.

- ( ) I used search less
- ( ) I used search about as often
- ( ) I used search more
- ( ) Not applicable/I never use search
- ( ) Don't know

My use of search (excluding Search Directed Navigation) changed as follows because of the availability of Search Directed Navigation: *

- ( ) I used search less
- ( ) I used search about as often
- ( ) I used search more
- ( ) Not applicable/I never use search
- ( ) Don't know

Comments

If Icon Highlights had not been available, I would have used Search Directed Navigation: *

- ( ) Less often
- ( ) About as often
- ( ) More often
- ( ) Don't know

If Search Directed Navigation had not been available, I would have used paid attention to Icon Highlights: *

- ◯ Less often
- ◯ About as often
- ◯ More often
- ◯ Don't know

Comments

Accuracy

Icon Highlight predictions seemed accurate *

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree
- ◯ Don't know/not applicable

I knew in advance whether Icon Highlights would highlight what I was looking for *

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree
- ◯ Don't know/not applicable

Search Directed Navigation results seemed accurate *

- ◯ Strongly agree
- ◯ Agree
- ◯ Neutral
- ◯ Disagree
- ◯ Strongly disagree
- ◯ Don't know/not applicable

Comments

Design

I found it hard to locate highlighted items when they were out of view *
For example, if you had to scroll in large folders.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree
- Don't know/not applicable

Comments

I found it harder to use Finder search because of Search Directed Navigation *

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree
- Don't know/not applicable

I got confused between Finder search and Search Directed Navigation *

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree
- Don't know/not applicable

Comments

The way items were highlighted with Icon Highlights was clear and intuitive *

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree
- ○ Don't know/not applicable

The way items were highlighted with Search Directed Navigation was clear and intuitive *

- ○ Strongly agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly disagree
- ○ Don't know/not applicable

Comments

Other

Do you have any other comments?

If you like, we can send you the results of this study once they have been finalised. This will likely be towards the end of the year.

- ☐ Please email me the results of the study when they have been finalised

Submit

Never submit passwords through Google Forms.

Powered by Google Drive

This content is neither created nor endorsed by Google.

Report Abuse - Terms of Service - Additional Terms