THE VISUALIZATION OF EVOLVING SEARCHES

a thesis submitted to

The University of Kent

in the subject of computer science

for the degree

of doctor of philosophy

By

Edward Suvanaphen

December 2006

Abstract

It is a common misconception that all web searches can be answered with a single query. It is true that when users have a clear idea of what they are searching for, they can specify an accurate and efficient query to the search engine and find pertinent results in the first 10 search results returned. However, studies of search engine usage by Jansen et al. ([56],[57], [59]) show that, on average, one out of every three users cannot satisfy their information need with a single search, and are forced to perform multiple searches.

This might be because they begin with limited information, or only a vague information need, and subsequently must browse their data, moving through a variety of different sources. Bates [6] theorized that as users search and reformulate their queries, they are affected by each new piece of information that they view, giving them new ideas and directions to follow. Their information needs are *evolving*. Furthermore, she noted that each new query contributed information towards the users final knowledge set, and as a result aided in fulfilling the information need. She postulated that a query is not satisfied by a single final retrieved set, but instead by different pieces of information gathered throughout the search. She called this the *berry-picking model* (an analogy to picking berries in the forest).

However, performing evolving searches can be time-consuming and difficult, and Bates outlined a set of interface issues that needed to be addressed in order to allow users to perform efficient evolving searches. As of yet no research has focused on the development of tools to directly aid users in performing evolving searches. This thesis will investigate the visualization aspects of evolving searches, looking at each of Bates's interface issues and considering different tool designs, techniques and technologies that would prove beneficial to a user performing an evolving search. The outcome of this thesis, is the design, implementation and evaluation of a tool for aiding users in performing evolving searches.

Acknowledgements

When I was ten, I told my father that I wanted to be a doctor. My father looked me in the eye, and asked me *"Why do you want to be a doctor?"*. I simply replied that I wanted to be just like him. If you asked me the same question now, I would reply *"It's complicated"*. I dedicate this work to my father, my first inspiration, the original Dr. Suvanaphen.

I would like to thank my supervisor, Jonathan Roberts for his advice and guidance, throughout the last four years. Jonathan has taught me many things and provided me with many opportunities in my research, and I am honoured to have worked with him.

Of my colleagues, I would like to thank Matthew Jadud and Rodolfo Gomez for their invaluable assistance in reading and providing comments on my thesis. I would also like to extend my thanks to all my office mates, Christian Jacobsen, Damian Dimmich, Keith Franklin, and Richard Walker, who have all supported me in my work at various points in my stay here.

I would also like to make a special mention to those people in Canterbury who have affected my life here. I would like to thank the people of Woodland way, Brad Wyble, Carrie Jadud, Romain Laborde, and Poul Henriksen for supporting me and putting up with my eccentricities. My special thanks goes out to Gustavo Herodier, long-time friend, fellow artist and partner in crime, who has had to put up with me the longest.

Finally i'd like to thank my family, for their love and support through out my life, and for providing me every possible opportunity. To my mother, thank you for each and every letter you sent, they kept me going when I was down, and gave me strength, when there was none left. To my father, thank you for constantly pushing me to be better, you were my inspiration, and the reason I wanted to become a doctor.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"You can't always get what you want, but if you try
sometimes, you might find you get what you need."
- Mick Jagger (The Rolling Stones).

Searching for information can be difficult, and try as you may you don't always find what you want. However, sometimes while searching you happen upon something that is different to what you were looking for, but at the same time interesting and relevant. Hence, you might "get what you need". From this perspective, it is easy to see that searching is a journey of discovery, and people learn things as they search which in turn may affect their needs, directions and goals.

These themes of discovery and changing needs are common to web searching. Take for example a standard online search : users will enter keywords into the search engine, in an attempt to find relevant information, viewing search results and browsing web pages as their session progresses. After viewing some results, users may change their minds and alter their topic of search. With every search, users adapt and change their behaviour based on the pages read and information viewed: they are performing an *evolving search*.

The **evolving search** and **berry-picking** model are both theories developed by Marcia J. Bates [6] which model the search behaviours of people seeking information in a library. Many authors have alluded to the validity of this theory, and have connected it to the search behaviour of people in online environments. Evidence from observations studies by Ellis [30], and Borgman [13], as well as work by O'Day et al. [86], support the existence of behaviour. As of yet no research has been focused on the development of tools to directly aid users in performing evolving searches.

This thesis investigates the visualization and user-interface aspects of evolving searches; it considers different information-seeking strategies, visualization techniques and interface methods

that could be used to support and help users in their effective evolving searches. The objective is to detail the design, implementation and evaluation of tools to aid users in performing effective evolving searches.

Examples and expressions

In this thesis, certain examples are used repeatedly to keep a level of consistency for the reader, as well as prevent any confusion that may arise. When an example is given of an evolving search in this thesis, it will either relate to planning a safari holiday in Africa, or 'the big five' safari animals (Africa's most dangerous wild animals). However, this is not to say that evolving searches are only performed when planning a holiday. Evolving searches are diverse, and can cover a large number of different subjects and situations, such as shopping online for a birthday present, or searching for a rare research paper online.

In this thesis certain non-standard expressions are used for specific reasons. When the search terms input into a search engine are stated in this text, a pair of square brackets are placed on either side of the search terms used. For example, the phrase [safari animals] would be used to represent a search where the words 'safari' and 'animals' were input into the search engine. This expression is used so that the reader is clear as to what constitutes a quotation ("safari animals") and what constitutes a set of search terms ([safari animals]). Some search engines use quotation marks to indicate that a set of search terms should be treated as a phrase. Using the square brackets expression, these can be represented without ambiguity e.g. when writing in this thesis about a search for the exact phrase "the big five" using the Google search engine [36], the phrase ["the big five"] will be used.

1.1 Motivation

Thousands of pounds are spent every year in an effort to develop more efficient and accurate search engines, and yet finding information on the World Wide Web (WWW) is still very difficult. The WWW is a vast resource, and searching through it is a massive task. If not for the advent of search engines, the Web would still be the domain of computer enthusiasts and academics, leaving many of the Web's resources untapped. Web searching can take one of two forms, *directed searching*, where users begin the search with a clear set of objectives, and *browsing*, where users begin with only a vague idea of what to search for. When users begin browsing, they lack enough information to specify an accurate or efficient query. The three most common situations that initiate browsing are a lack of initial information, searching in a new area, or a lack of direction.

Sometimes users try to recall a piece of information but can only remember fragments of

	AltaVista.com		AllTheWeb.com		Excite.com	
	1998	2002	2001	2002	1997	1999
Searches performed						
1	77.6%	47.6%	53.0%	59.0%	48.4%	67.0%
2	13.5%	20.4%	18.0%	16.0%	20.8%	19.%
3+	6.9%	32.0%	29.0%	25.0%	30.8%	14.0%
Results pages						
1	85.2%	72.8%	83.0%	76.0%	28.6%	58.0%
2	7.5%	13.0%	10.0%	13.0%	19.5%	19.0%
3+	7.3%	14.1%	7.0%	11.0%	51.9%	23.0%

Table 1.1: **Web log data : multiple searches.** Results from multiple studies performed by Jansen et al. [57] [59] , showing that a significant portion (between 30 and 50%) of users perform multiple searches.

	number of queries per session	
	2	3+
Total sessions	254	483
Multi-tasking sessions	206	441
% of Multi-tasking sessions	81.1%	91.3%

Table 1.2: **Multi-tasking in web searches.** Spink et al [104] performed an analysis using the aforementioned AltaVista 2002 data, investigating multi-tasking in web searches.

the details. This lack of initial information limits their search vocabulary, and restricts the effectiveness of the queries they generate. For example, a situation where users view an advert for safari holidays, and then several weeks later decide to go on a safari holiday, but are unable to recall many of the details of the website or the holidays shown. Sometimes users investigate a new area, where they have little or no knowledge, limiting their search vocabulary. An example would be if users searched for information about a country they wished to visit, but had never visited before. Sometimes users have not decided what they are searching for. For example, when picking a destination to go on holiday, users might have a general idea of what type of holiday they want (e.g. by the beach, at a ski resort), but they might not know exactly which country or town will suit their requirements best. They then need to browse websites to gather enough information to make an informed decision.

To construct an adequate query in such situations, users must start by generating very broad and general searches based on the limited information available, and absorb information as they progress. The information from these *multiple searches* is then incorporated into future searches, bringing them closer and closer to a more optimal set of search terms.

Multiple searches are often the result of poor query formulation. Jansen et al. [57] observed multiple search behaviour in a series of studies of search engine logs. Table 1.1 shows a selection of results from Jansen's studies. The results show that while the majority of searchers will only engage in a single web search on, between 30% and 50% of users will perform multiple searches. A follow on study by Spink et al. [104] investigated the occurrences of multi-tasking

within web searches, (see Table 1.2) The results seem to indicate that the search topic would be changed at least once in a large portion of multiple search sessions (81% in 2-query search sessions, and 91% in 3 or more query sessions). This evidence suggests that multiple searches promote *information diversity*, subtly changing a searcher's topic as the search progresses. No evidence has been given to suggest exactly why people change topics during a search, however possibilities include hidden associations between topics, search term ambiguity and impulsive search behaviours.

If different searches performed by users have hidden associations, it may appear as if users are changing their topics during a search, when in fact they are still researching information on the same topic. For example, hotels and safari animals would be considered as different topics unless used in reference to a tourist's visit to South Africa, where the tourist would be interested in investigating both topics.

An initial search performed by users can sometimes prove too ambiguous, and careful inspection of the first set of search results may result in a more concentrated (yet topically different) search. For example, when searching for the 'big five' safari animals, the top five safari animals in Africa, users may perform the search [the big five]. However, 'the big five' is a very ambiguous phrase, and also refers to the five main psychological models, the top five record labels, and a chinese text character set.

As users refine their searches, it may seem as if a topic change is taking place. Searching in one topic sometimes triggers something completely unrelated, for example, while searching a news website for information on safari animals, users may happen upon an article for healthy eating that may interest them, and thus change topic.

Multiple searches, browsing and information diversity are traits that appear in certain types of web searches, and often cause users difficulties, due to the large amount of information generated while searching, and the large amount of time needed to process the information. These traits form part of the evolving search model, and these 'difficult searches' are very often evolving searches. By contrast, evolving searches are not the only type of search users perform, and most of the time users will perform 'easy searches' where the user inputs a set of search terms, and immediately receives the information that he desires in the first set of results returned.

1.2 The evolving search model

Over a decade ago Marcia J. Bates suggested that while browsing information, users develop new ideas and directions, *evolving* their information need as they forage for information. She named this the **evolving search** [6]. A detailed model of the theory can be seen in Figure 1.1. Bates also theorized that the final answer to an information need was not formed of a single set

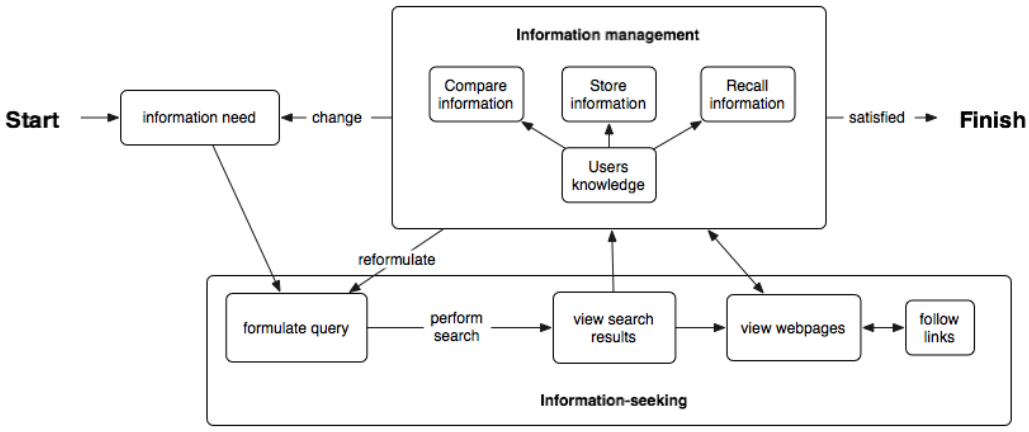

Figure 1.1: **The evolving search model.** The model was invented by Bates [6], and gained popularity as a more accurate view of web searching behaviour than previous models based on traditional information retrieval. The evolving search takes into account all user information processing actions during the search (signified here by the added 'information management' section), and unlike its predecessors allows the option for users to modify their starting information need.

of results, but was made up of several different pieces of information, obtained from different sources at different stages in the search. She named this the **berry-picking model**. The berry-picking model is analogous to the picking of berries in a forest. The berries are often scattered around and the picker is forced to go from bush to bush picking one set of berries at a time. In the same way, the information needed to answer an information need is sometimes scattered among many different web sites, forcing searchers to move from web page to web page gathering information.

Bates's theories were supported by several authors (Line [68], Hogeweg-de-Haart [48], Stoan [107] and Stone [108]) and in a variety of different environments, particularly in the social sciences and humanities. Ellis's [30] work on social scientists supports and amplifies the results of earlier studies, and Kuhlthau's work [64] with high school students suggests that there is a great deal of exploratory searching that goes on, both before and after a topic for a paper is selected. Both Mann [71] and Hearst [43] cited Bates's model as a more realistic view of the web search process, and both used it as a basis for their own search behaviour models. Most recently O'Day et al. [86] confirmed the existence of the berry-picking model in an experiment which observed the search behaviour of professional intermediaries. During the experiment it was noted that clients would, over time, conduct a series of diverse but interconnected searches on a single, problem based theme, rather than one extended search session per task. O'Day noted that it was often the accumulation of the search result information, not the final set of search results, that had the most value for the library clients. This became apparent when clients began writing summaries of the materials they had found, after the search was finished.

Figure 1.2 details an example of an evolving search, based around a person planning a safari

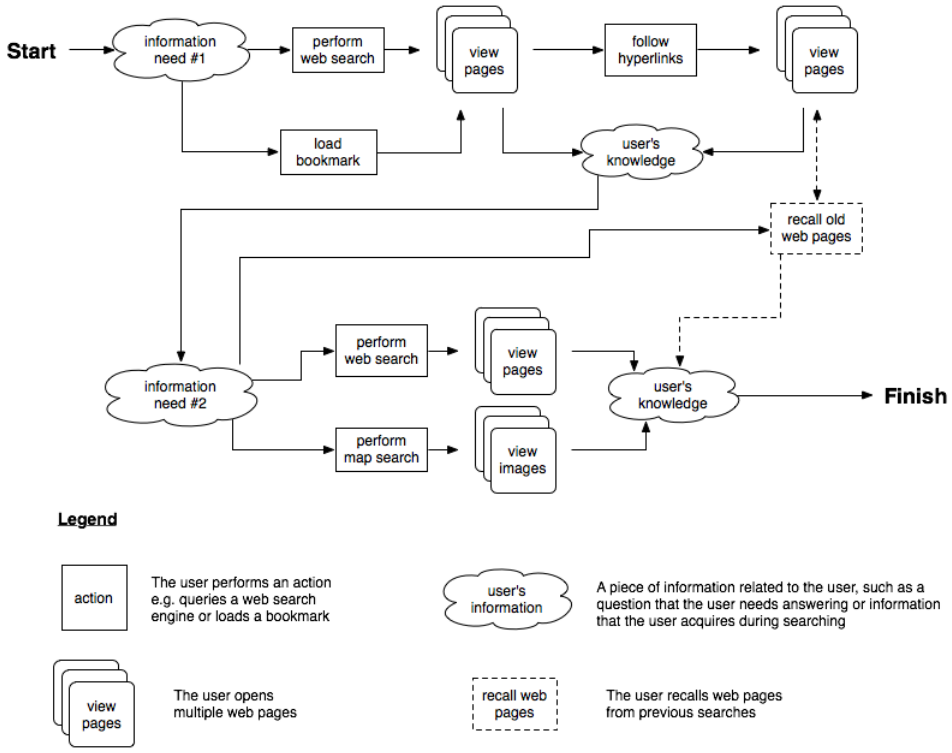

Figure 1.2: **The berry-picking model : an example** In this scenario a person is planning a safari holiday, and searches for cheap holiday packages in africa, which include safari tours (*information need #1*). The person gathers information from various sources (websites, searches, bookmarks), storing and comparing the information as the search progresses. At some point during the search, the person reads about the 'big five' safari animals, and becomes determined to see them. This changes the users information need from searching for cheap hotels to finding out which African countries have the 'big five' (*information need #2*). The person recalls some sources, performs new searches and saves the data.

holiday (see Figure 1.2). The person's initial information need is to look for cheap holiday packages which include safari tours. First a web search for websites offering safari holiday packages are made and compared, to find the cheapest package. The person also loads a bookmark of a web page from a previous web search, which advertised cheap holiday packages. As the person views web pages and gathers useful information, he learns about the 'big five' safari animals, the five most dangerous wild animals in Africa. The person becomes interested in the 'big five' and performs a second search, this time concentrating on finding countries in which the 'big five' can be seen. The person's information need has now changed from finding cheap holiday packages to finding countries with the 'big five'. The person will view additional web pages, and also perform a map search using an online map service to find the geographical location of countries that he reads about. The person finds that the 'big five' can be seen in South Africa, and he consults the web pages from his previous search (recalling them) to find out if any of the cheap holiday packages he discovered in the first search go to South Africa. The search ends

when the person has found a suitably cheap safari holiday package going to South Africa.

This example demonstrates that the information needs of users change as search queries change, and that users refer to multiple sources, and sometimes need to refer back to previously viewed information (such as webpages, bookmarks, or previous searches). While many of Bates's observations were based on the searching habits of people consulting manual sources (libraries) and users interactions with databases, she was a great advocate of the then blossoming hypertext environment describing it as 'tailor-made' for solving many of the problems of the evolving search.

Bates's interface issues

In Bates's research [6] she detailed interface issues that needed to be addressed in order to improve the efficiency of users performing evolving searches. These issues were related to designing interfaces in a manner that would allow users to search more naturally as well as aid them in finding information. There are several different strategies that are called upon when searching in the library (e.g. footnote chasing, journal run, area scanning) and Bates noted that interfaces should enable easy switching between each of the different search tactics. She also stated that an effective online environment must allow users to browse quickly and randomly over the information, much like the physical act of scanning over the titles of books on a shelf, or skimming through the pages of a book.

Bates suggested modeling the search interface after a real-life analogy in order to facilitate understanding of the interface and guide the actions of users. She suggested that it may be the physical layout and organization of certain familiar search structures (such as libraries) that people are most familiar with, rather than the complex intellectual relationships developed with catalogue entries, books, periodical indexes, journals etc. Thus, creating a virtual physical layout on the screen may make it easier for users to interact with information. Observations by Bates showed that searchers in the library would consult multiple sources of information (e.g. library databases, citation indexes, journals) as well different types of text (e.g. narrative, statistical, bibliographic references). The limitation of this strategy was that users had to physically relocate themselves each time they wished to consult a new data source. Even switching between different databases (during Bates's period) was difficult. Ideally users should have quick and easy access to different sources and types of data.

Bates also noted that, under the berry-picking model, evolving searches generate a large amount of information, from diverse topics. As a result it is important that users are able to view the data as a whole, as well as selectively inspect individual aspects of the data. It is also important that users be able to manage the diverse strands of multiple searches generated, and keep track of the information generated. Given the large amount of information generated, it is

also necessary to be able to highlight or flag interesting pieces of data in temporary storage, for specific viewing later. Similarly users may wish to recall information that they had previously visited, signifying the importance of an automatic storage system, that allows quick and easy recall of data.

The evolving search model was originally based on observations of searchers in the library, as well as their interactions with databases. As a result, some of Bates's evolving search issues do not apply to searchers on the WWW due to differences in technology and information structure. These differences have also led to the introduction of new interface issues, which will also be looked at in this research.

1.3 Contributions

Evidence based on the observation of online search behaviour (in section 1.1) alludes to the existence of evolving search behaviour in searchers of the WWW. As of yet no research has been focused on the development of tools to directly aid users in performing evolving searches. The objective of this thesis is to detail the design, implementation and evaluation of tools to aid users in performing effective evolving searches. The research will aim at addressing five evolving search interfaces issues, three taken directly from Bates's interface issues [6], and two new interfaces issues, not present in Bates's observations, which are related directly to searching in an online environment. The three issues taken from Bates's work [6] relate to the areas of information visualization, opportunistic searching and information recall.

The issue of **information visualization** relates to the large amounts of data generated from searches. Evolving searches in an online environment generate multiple searches and multiple Web pages, which make viewing the data as a whole difficult. Visualization techniques exist that allow users to view and seek specific information in such circumstances. The issue of **opportunistic searching** relates to the common search tactic of browsing randomly through data. Existing search engine interfaces limit user's abilities to search opportunistically, because of existing display conventions (textual format, displaying 10 results at a time, needing to use the 'next' button to view more results). Existing search interfaces would benefit from the application of visualization techniques that allowed opportunistic browsing. Bates identified that **information recall** was also an issue, since it is very easy to lose track of specific pieces of data amongst the large quantities of data generated. The same can be said for web searching, where users often have a hard time recalling previously visited web pages and search results. Various visual recall and history tools can be introduced to interfaces to aid users in recalling data.

Some interface issues are not mentioned in Bates's work [6] due to differences between searching in manual print environments (libraries) and online environments (WWW). These are issues related to seeking interesting and relevant information, and managing user's information and search tools. Search interfaces should provide different ways of **seeking interesting and relevant information**, as well as ways to display the information from different perspectives. Visualization techniques can be used to display different aspects of the data and meta data of search results as well display any correlations or subsets that may arise, which users would normally have to achieve cognitively. During an evolving search, **the management of data** becomes important because of the numerous threads generated as user's information needs change. It is important to keep track of these in case users should find themselves back tracking, and wanting to follow previously visited threads. Tools to manage this data can be beneficial in recalling previous information as a basis for exploring new areas and finding more interesting information.

This research will make the following contributions: (1) provide an insight into the problems of evolving searches, and detail the different visualization techniques that can be used to solve these problems, (2) implement a working evolving search tool, based on the research, and (3) design and implement an experiment to analyze the viability of the design theories, as well as the effectiveness of the visualization techniques in aiding users to perform evolving searches.

1.3.1 Publications

Throughout the course of the research a set of papers were peer-reviewed and published for different conferences. The goal of the research was to design and implement a tool to help users perform evolving searches. Each of these papers followed the development of an evolving search tool, starting from its basic components, up to the final design.

The paper *"Visual bracketing for web search result visualization"*, looked at a display and interaction component of the evolving search tool, introducing a novel focus-and-context methodology for displaying and browsing web search results called 'visual bracketing'. The paper was presented at the IV03 (information visualization) conference in 2003, and was published in the conference proceedings.

The paper *"Textual difference visualization of multiple search results utilizing detail in context"*, describes the SES (Search engine similarity) tool, which introduces a method of finding and displaying similar results across multiple searches. The paper was presented at TPCG (Theory and practice of computer graphics) in 2004 and was published in the conference proceedings. This was followed by the paper, *"Explicit verses Implicit: An analysis of a multiple search result visualization"*, which described an experiment to test the effectiveness of the SES tool. The paper was presented at the IV04 (information visualization) conference in 2004 and

was published in the conference proceedings.

A final paper was written entitled *"Visualizing evolving searches"*, describes the design, implementation and evaluation of the final design of the EvoBerry evolving search tool. This paper was submitted to the VAST06 (Visual Analytics) conference, and is awaiting review.

1.3.2 Chapters

In this chapter the theory behind the research has been discussed, as well as the aims for this thesis. Below is a summary of the structure of the thesis, as well as the contents of the chapters:

- **Chapter 2** discusses the theories and models that surround web searching, and how they relate to the evolving search.
- **Chapter 3** covers the technology and visualizations which can aid users in performing evolving searches. The chapter will look at the current state of web searching technology, and then go onto discuss the different interface issues presented by Bates, as well as the techniques used to solve them.
- **Chapter 4** describes the design of the evolving search tool and presents the various stages of implementation that took place on the way to developing the final tool.
- **Chapter 5** presents the final design of the EvoBerry search interface. A short summary of the technical challenges faced when developing the tool is also included.
- **Chapter 6** details the design and implementation of an evolving search experiment. This design is used in turn to test the effectiveness of the EvoBerry interface.
- **Chapter 7** analyzes the data returned from the experiments. Trends in the data are identified and explained.
- **Chapter 8** concludes the thesis and discusses future work and research.

Chapter 2

Model and Theories of Searching

"Search is actually a last resort. If you knew the answers you wouldn't need to search."
- John S. Rhodes (webword.com)

Web searches are initiated when users find a deficit in their knowledge which cannot be satisfied easily by consulting information sources in the nearby environment (e.g. asking friends, consulting nearby documents). This 'failure' to acquire information becomes an *information need*, which consists of the specific elements users need to acquire in order to fill the gaps in their knowledge. One of the crucial differences between information-seeking and information-retrieval is the specification of the information need. In information-retrieval it is assumed that users have a clear idea of what their information need is, and can specify exactly what they want, and where to find it. In information-seeking, users have a less clear idea of their information need and often must 'browse' in order to gather sufficient information to build a clearer information need.

Belkin [10] called this the *Anomalous State of Knowledge* (ASK). The theory behind the ASK hypothesis is that an information need arises from a recognized anomaly in the state of knowledge of users, concerning a topic or situation that they are unable to specify precisely as a query that can be understood by the search engine. This inability to specify a precise query manifests itself in the form of multiple searches, browsing strategies, and reformulation based on information acquired during the search, all of which are themes discussed in this chapter.

In this work, web searching refers to the act of using online search engines to search the World Wide Web (WWW) for information. Users begin by typing in a set of words (the *query*) into the search engines interface, and are returned a list of search results (typically each page contains 10 results), each search result containing a short description and a link to a website that the search engine thinks matches the search criteria. The technical aspects of search engines

and web searching, will be discussed in more detail in Chapter 3.

There are two basic forms of web searching behaviour that form the basis for most web searching strategies, *directed searching*, and *browsing*. In directed searching, users have a clearly defined objective (e.g. find out the cost of a flight to South Africa) and have an established vocabulary that can be used to return a set of pertinent search results (e.g. using the terms, [cost], [flight], [south africa]) from the search engine. Directed searches only require a single search before users find the information they need. In browsing, users will start by performing very broad searches, and will spend a lot of time reading web pages, gathering information that may aid them in narrowing their search. This is either because they are unable to specify their query clearly, begin the search with little or no knowledge of the domain they are searching, or are undecided as to what their information need is. These are both considered extremes on a spectrum of web searching behaviour, and very often a user's search will consist of a combination of the two behaviours utilized at different points in the search session.

Sometimes users have a clear idea of what they want, but don't know where to find it and cannot translate the concepts in their head into words that will form an effective search. This will be referred to as *browsing because of limited information*. For example, when reading web pages online users might happen to see a website advert promoting cheap flights to South Africa, but only give the advert a cursory glance, taking in the colours, objects and pictures of the advert. But when users try to recall the advert later, they will be hard pressed to find the advertised website on the World Wide Web, with the few details remembered. They will attempt a few broad searches based on what few details they can recall, and gather information from the results returned in the hope of finding a link to the sought website, or more keywords that will help generate a search which is a closer match.

Users might be searching for general information in an area in which they are unfamiliar, and thus be less concerned about what information they obtain or where they obtain it from. This will be referred to as *browsing because of unfamiliar territory*. In such situations users have little or no knowledge of the area, and are merely searching for a piece of information to act as a stepping stone to further information. For example, when researching a subject or area with which they are unfamiliar, users will perform several broad searches, and then begin visiting numerous pages on the subject.

Sometimes users might start the search without a clear information need; they have only a vague idea of what they are looking for, and don't know where to start looking. This will be referred to as *browsing because of unclear objectives*. Most of the time users will either 'know it when they see it' or will get a better idea of what they are looking for as their search progresses. For example, when planning a holiday, users might begin with a vague information need ("a cheap holiday"), and thus will want to generate as much information as possible. By generating

and browsing the information, users begin to form a concept of what their information need is, and start forming a clear set of objectives.

Browsing strategies often have no clear end-criteria (e.g. an objective, a reason to stop), and hence tend to take far longer than directed searches, generating multiple searches and web pages. These two strategies bear much relation to the two *guided search* activation components, *bottom-up* activation and *top-down* activation. According to the theory of guided search by Wolfe et al. [124] (based on Treisman's [115] feature integration theory), when users search for something, their brain generates a feature map, where 'activation' on the map generates an area of interest that users will wish to visit. As stated, activation on this map can occur by either *bottom-up activation*, when an item is strikingly different to its background or context, or by *top-down activation*, which occurs when the user's search is based on a single feature that is used to designate potential targets from distractors. When users are browsing, it is assumed that they are using bottom-up activation, scanning randomly through web pages until something differentiates itself from the irrelevant information, 'popping-out' into the field of view. When performing a directed search, it is assumed that users will search via top-down activation, since they know exactly what information to search for, and will differentiate the data based on that feature. While directed searches and browsing, are the most common forms of searching, they are by no means the only ones; Wilson [122] discussed four different forms of searching as part of his model of information-seeking behaviour:

- **Active search:** This is the same as the *directed search*, where users are actively seeking out information, and is the most common form of search.
- **Passive search:** This occurs when a search (or other behaviour) results in the acquisition of information that happens to be relevant, which will in turn trigger a new thread of search.
- **Ongoing search:** Also known as a *monitoring strategy*, this occurs when a search is performed repeatedly on the same subject, to keep users up-to-date on changing information related to the subject area.
- **Passive attention:** Although not strictly a form of searching, it was included as part of Wilson's search and acquisition strategies. This happens when users acquire the information without actively seeking it out, such as when listening to the radio or watching the television.

Web searching is sometimes described as 'looking for a needle in a haystack'. This analogy was adapted by Koll [63] as part of his description of web searching strategies, where each strategy was considered a 'needle in the haystack' problem. Below are the first four strategies:

- **A known needle in a known haystack.** Similar to the *directed search*. Users are familiar with the search territory, and know exactly what to look for and where to find it.
- **A known needle in an unknown haystack.** Similar to *browsing because of limited information*. Users have a clear idea of what they want, but not of where to find it.
- **Any needle in a haystack.** Similar to *browsing because of unfamiliar territory*. Users are less concerned about the characteristics of the particular search result they are looking for.
- **An unknown needle in an unknown haystack.** Similar to *browsing because of unclear objectives*. Where users only have a vague idea of what to look for, and don't know where to start looking.

The strategies in Koll's list bear similarities to previously discussed strategies, covering the directed search as well as all three instances of browsing strategies. Koll [63] lists several other 'needle in the haystack' strategies, which are more specific in purpose:

- **The sharpest needle in a haystack.** Users have identified a specific variable (e.g. number of images, number of links) that is of great interest to them, and are only concerned with retrieving the result with the largest quantity of that variable.
- **Most of the sharpest needles in a haystack.** Like above, users are looking for information revolving around a certain variable, but in this case they search for multiple sources.
- **All the needles in a haystack.** Users are still searching for results based around a piece of information, but wish for ALL sources. Such an expansive search can take a long time, if users are very motivated to find all the information.
- **Affirmation of no needles in a haystack.** Users want confirmation that a piece of information does not exist in the result set. Due to the dynamic and ever changing nature of the web, this is difficult to perform and not always guaranteed to be correct.
- **Things like needles in a haystack.** Users search for objects which are similar to their original target.
- **Let me know whenever a new needle shows up.** This form of searching bears similarities to monitoring strategies. Users of the WWW will often return to regularly updated websites to view new information (e.g. weblogs), or utilize automated technology to keep up-to-date e.g. RSS and podcast technology. Users will also perform searches

multiple times over a long period of time, in order to keep up-to-date on information surrounding a particular subject.

- **Where are the haystacks?** Users may not know where to start looking for information. For example, a student without prior knowledge would not know which website to go to find research papers and thus must first perform a search to acquire this information before performing their principle search.

- **Needles, haystacks - whatever.** Oddly enough this is a recognized search behaviour. There are times when users will follow a link or scrap of information that is interesting, but not related to the current information need at all.

These examples show that while search models often share a basic set of strategies (directed searches and browsing strategies), they are also diverse in their interpretation of search behaviour, and varied in the level of granularity at which they study search behaviour. This chapter will look at the different search strategies more closely, as well as detail many of the search models developed to explain users search behaviour.

The *information-retrieval* model, the original concept behind seeking information in online sources, will be discussed in the next section. The concept of *information-seeking* will be introduced, as well as the various models used to describe this form of search behaviour. The chapter finishes with a discussion on the differences between these models and the evolving search, as well as a look at studies which have been performed to capture the user's search behaviour.

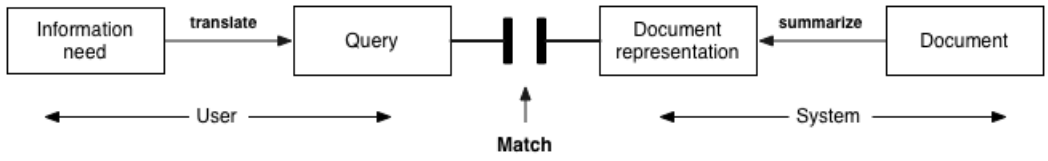

Figure 2.3: **A model of classic information-retrieval.** This model is based around users interactions with databases (as seen in Bates [6]). The underlying theory is that users (on the left) will translate their information need into a query in the systems language, that will return them an exact match for the query provided.

2.1 Information-retrieval

In the literature, the term information-retrieval has sometimes been used synonymously with information-seeking. Since most modern information-seeking methods are derived from information-retrieval methods they both share the same premise (the search for information in an electronic environment) and for the most part they share the same interface (users must translate their information need into a string of text that is input into the system), from which a list of

search results are returned. But this is where the similarities end, as Jansen et al [57] stated *"Internet searching is very different than information-retrieval searching, as traditionally practiced and researched. Internet IR is a different IR"*. Information-retrieval is typically associated with database interactions, sometimes referred to as 'Classic information-retrieval' or simply IR. Figure 2.3 describes the classic model of information-retrieval where, users translate their information need into a query, understandable by the system, which is then submitted to the search engine. The system then attempts to match the query to a document representation in the database index, and returns a list of possible matches.

This concept was designed with the assumption that users would be trained to use the interface to communicate with the database effectively, as well as know exactly which piece of information they wished to retrieve. Thus it was assumed that a single query would suffice for each information need. While this model was established with well founded principles, over time it began to display inaccuracies and inadequacies, due to changing technologies, and better understanding of user's search behaviour. One of the questions asked at the time (in [87], and [10]), was *"why is it necessary for users to represent their information need in a form understandable to the system?"*. This was a reference to the fact that databases often required knowledge of specific query languages for interaction. It was established that it would be more efficient to represent the query the way it was stored in the user's head, or as plainly as users would describe such an information need to a fellow human being. The two processes of information-seeking and information-retrieval differ in other significant ways such as the structure of the query, the technology used and interaction behaviours used.

Information-retrieval operates on a single query approach, where a single correct query will match a representation in the database and return a single set of results that will fulfill the information need. Information-seeking is rarely so simple, and it is not uncommon for users to query the search engine multiple times. Information-retrieval queries typically range between 7 and 15 words per query. Web searching however typically averages between 1 and 4 words per query [57].

Changing technologies also ensure significant differences in the two models. Information-retrieval is associated with database interactions, and thus performs matches using an index system, where the query is only checked against the document's abstract or keywords. By comparison, most web search engines scan the entire document. Information-retrieval in databases rely on searches of pure boolean logic, whereas web search engines use statistical ranking by default, with options for applying additional boolean operations. One of the most significant differences is the absence of hyper-linking in information-retrieval systems. Hyperlinks (also known as links, web links or anchors) are one of the key elements used in web searches, allowing users to continue searches beyond the returned results by following links to similar pages.

This increases the range of available data, and increases the chances of finding relevant information. However, in information-retrieval, once a document has been returned and retrieved, that particular thread of searching ends.

In the past it was not easy to move between different collections using information-retrieval systems, because of the time, knowledge and effort needed to switch between databases. If a website is considered a collection of information, then the World Wide Web is effectively a multitude of different collections of information, all easily accessible from one single interface, making switching between collections a trivial task. This gives way to more varied and spontaneous searching, as well as providing a larger pool of interesting information.

A commonly held theory of information-retrieval is that more precise and narrower searches will generate smaller results sets and thus more accurate results. Eastman [28], in an experiment involving students searching the web, showed this to be a less reliable method than first thought. Her theory stated that, given the number of results often returned by a search engine (between 300 and 30,000 in her example), reducing the result set size was not as important as improving the quality of the top listed items, as users were unlikely to look at all 300 results let alone 30,000. Databases often require specific knowledge of a query language (e.g. SQL) to access and interact with the system, whereas search engines employ natural language representations for their queries, enabling people to use them with little or no prior knowledge of the system. This has been less of a problem in recent years, since database interfaces have become more user friendly, with a move towards form-filling and graphical interfaces.

The single greatest difference between information-retrieval and information-seeking strategies is the inclusion of hyperlinks. The addition of the hyperlink structure allows users to travel beyond the returned list of search results, and 'browse' other linked web pages, a feat not possible in information-retrieval. The ability to browse information, following it opportunistically and exploring diverse and sometimes unrelated paths, is of benefit to users when their information need is not fully realized. Browsing strategies form an important part of the information-seeking process, and will be discussed further in the next section, with relation to the different theories and models related to information-seeking.

2.2 Information-seeking

In this thesis, the term 'information-seeking' is used to describe the process by which users search for information on the World Wide Web. Most often the term information-seeking is used with reference to web search engines, although it is understood that interactions with the search engine are not the sole interactions in an information-seeking episode, and that the information-seeking process starts before and continues after interactions with the search

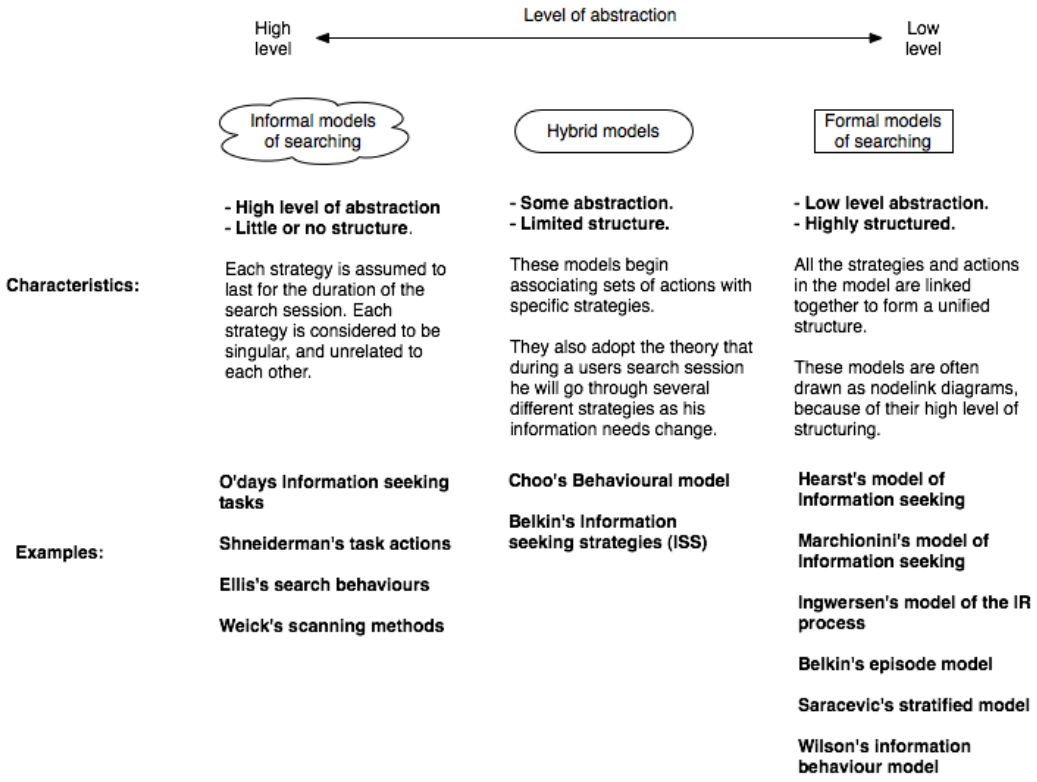

Figure 2.4: **Models of searching:** This table shows the different search models discussed in this section. The models are ordered by level of abstractions, on the left are the *Unstructured models* which operate at the highest level of abstraction, on the right are the *Structured models* that operate at the lowest level of abstraction, and in the middle are the *Hybrid models* which share some features of both models.

engine. The strategies used and actions performed in an information-seeking episode are wide and varied, hence the need for several different models to explain each of the different theories of search behaviour. The works on search behaviour can be divided into three sub categories, the higher level *unstructured search models*, lower level *structured search models*, and the *hybrid search models* that share some traits from both extremes (see Figure 2.4).

2.2.1 Unstructured search models

Information models represent the highest level of abstraction. They consist of a set of unconnected strategies, where it is assumed that a single strategy fully describes the actions of users during a search session. Such models can be described by very simple static strategies, such as the aforementioned *directed search* and *browsing* strategies. These two strategies occur in nearly every theory described in the research, although sometimes the *browsing* strategy is broken up into more distinctly defined sub-strategies (as demonstrated in the introduction). A comparison of three basic unstructured models by O'Day [86], Shneiderman [100] and Weick [25] can be

Models	Directed search	Browsing	Monitoring	others
O'Day's *information-seeking tasks*	following a plan	exploring a topic	monitoring a topic	-
Shneiderman's *Task actions*	specific fact finding	open ended browsing	-	extended fact-finding, exploration of availability
Weick's *Scanning methods*	formal search	undirected viewing	-	informal search, conditioned viewing

Table 2.3: **Table of unstructured search models.** This table compares the components of the search models of O'Day [86], Shneiderman [100], and Weick [25]. The three most common strategies are represented here (directed searching, browsing and monitoring). The table shows which of these strategies exist in the discussed search models, and what they are called in that model. Here it can be seen that most search strategies share the similar behaviours of *directed searching*, *browsing* and *monitoring*.

seen in Table 2.3, showing how each model maps to the previously mentioned search strategies of directed searching, browsing and monitoring.

O'Day et al [86] described three different search modes, *following a plan*, *exploring a topic* and *monitoring*, based on their experiment which studied the information-seeking behaviours of professional intermediaries. The first two correspond to the directed search and browsing strategies respectively. The *monitoring* strategy is when users will repeatedly visit the same web page over a period of time, to check for up to date information, similar to the aforementioned 'ongoing search' by Wilson [122]. Shneiderman [100] elaborates on the basic strategies in his set of *task actions*. His 'specific fact-finding' and 'open-ended browsing' strategies correspond to the afore-mentioned directed search and browsing strategies (also seen in Table 2.3). Shneiderman's 'extended fact-finding' operates on the theory that users have a well defined query, but have not established a clear set of end-criteria. The 'exploration of availability' strategy (like open-ended browsing) is an unstructured approach, but involves investigating an area or context that the users are not familiar with.

Looking at Daft and Weick's [25] model, four scanning modes can be seen, *undirected viewing*, *conditioned viewing*, *informal search* and *formal search* (see Figure 2.5). Undirected viewing is much akin to browsing, the information need is not well specified, and the overall purpose is to scan broadly over the information. Different information sources are used and large amounts of data are screened. Daft and Weick's [25] next scanning mode is conditional viewing, where users begin to focus their search, selectively directing their attention to specific sections of information so that they may evaluate, acquire and learn the information. Although still a broad strategy, it focuses the search on all information related to a particular theme or topic. The third scanning mode, informal search, becomes even more focused than the previous strategy, actively selecting pieces of information that will aid in satisfying the information needs of users. The final scanning mode, formal search, closely resembles directed searching, where users make a deliberate or

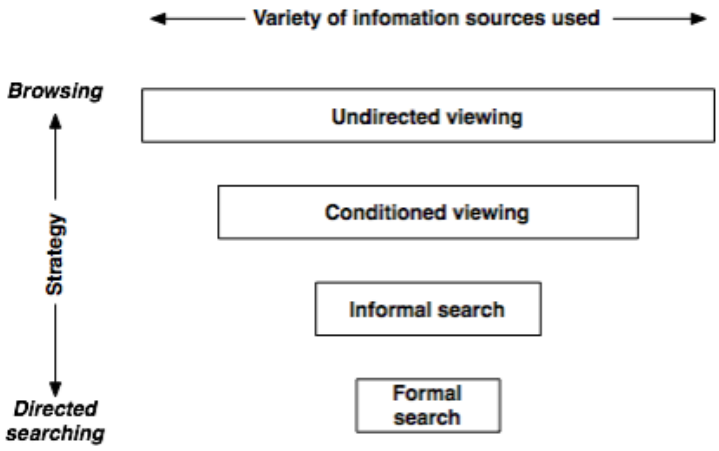

Figure 2.5: **Daft and Weick's scanning modes ([25]).** These scanning modes can be ordered both in terms of the breadth of the search (the number of different information sources used), and by which strategy it most closely resembles. The number of information sources used is represented by the width of the bar, and its vertical position in the list denotes how closely it resembles the two strategies, directed searching and browsing.

planned attempt to obtain specific information on a clearly specified issue. The search is formal because it is structured according to some pre-established procedure or methodology. However, not all unstructured models operate at such a high level. Ellis and Haugan [31] added a more clearly defined structure to their search model, based on the observed behaviours of social scientists, research scientists and engineers. In this model, the strategies are defined as different parts of the search which can be put together to form an information-seeking session (seen below).

- **Surveying:** Also known as 'starting', this activity signifies the beginning of a search. Ellis and Haugan specified that surveying was used for (1) approaching a new topic, within an unknown field, (2) finding out what has been done in that field earlier, (3) identifying the current status of that topic, and (4) finding background information for new elements of a project. The start is characterized by an information need.

- **Chaining:** This involves following chains of referential connection between sources, in order to identify new sources. Chaining can be performed backwards, using an article as a starting point and then following references from there, or forwards, utilizing a citation index to find works that reference to that article. This can easily be related to hyper-linking in web pages, which operates in much the same fashion, backward chaining being normal hyper-linking.

- **Monitoring:** This involves maintaining awareness of developments and technologies in a field through regularly following particular sources. At a very basic level, it can involve

simply revisiting websites of interest regularly, but also covers site update tools such as RSS, push technology and pod-casts.

- **Browsing:** Much like the previous definition of browsing, this strategy is casual and undirected. This can be at a high level (browsing search results) or at the low level (browsing the contents of several web pages).
- **Distinguishing:** Users, based on their own preset parameters, attempt to rank each piece of information recieved, so as to create a hierarchy of importance. When viewing search results, the search engine implements an automatic ranking of results, effectively distinguishing results for them, but users still inspect the results returned and distinguish their own levels of importance to each result based on the given metadata. Bookmarks can be used to help distinguish results of particular interest.
- **Filtering:** This activity involves users subconsciously selecting a set of criteria with which to filter the search results, and weed out uninteresting results. Most users perform this operation automatically when web searching.
- **Extracting:** Once a few interesting results have been distinguished from the set, the information is then extracted. In web searching this can take the form of reading a set of selected web pages or documents from search results in the list.
- **Ending:** The operations that need to be performed at the end of the search e.g. storing the data found, in a format that can easily be retrieved (saving documents).

With the exception of *Starting* and *Ending*, an information-seeking session can be comprised of any number of these strategies in any order. Although Ellis and Haugan's [31] model moves closer to defining the lower level actions, they have not yet imposed a rigid structure, allowing a more flexible model. Such models will be referred to as *Hybrid search models* because they combine the flexibility of Unstructured search models with a small set of defined actions.

2.2.2 Hybrid search models

Choo et al. [20] developed a Hybrid model, based on the previously mentioned model of Daft et al. [25] (the four scanning modes) and the strategies detailed by Ellis et al. [31] (in the previous section). By combining these two models, Choo was able to build a table of four search strategies, each of which is associated with a set of search actions that take place when using that strategy. This can be seen in Table 2.4.

	Starting	Chaining	Browsing	Differentiating	Monitoring	Extract
Undirected viewing	Identifying start pages	Following links	-	-	-	-
Conditioned viewing	-	-	Browsing pages	Bookmarking, Printing	Revisiting bookmarks	-
Informal search	-	-	-	Bookmarking, Printing	Revisiting bookmarks	Search engines
Formal search	-	-	-	-	Revisiting bookmarks	Search engines

Table 2.4: **Choo's behavioural model of information-seeking on the web.** By combining the models of Daft [25] and Ellis [31], Choo [20] was able to define a set of search strategies which had associated search actions e.g. Using an undirected viewing strategy, a searcher will often perform starting and chaining actions, such as identifying start pages and following links.

Starting at the top-left corner, users begins their search in a mode of *undirected viewing*, starting without a specific information need in mind, and browsing the subjects broadly. The associated action *starting* refers to users beginning their search, either by inputting a query into a search engine, or traveling directly to a URL specified from an external source. While in undirected viewing mode, users may follow hyperlinks from the first set of pages that they view, thus beginning to acquire information through *chaining*. Moving down the table, in *conditioned viewing* users begin to narrow their search down to a few key topics, and then begin *browsing* the data. At this point users might also begin *differentiating* pages of interest (through bookmarking, copying links and printing pages), so that they can be revisited and re-examined in the future.

Moving down the table once more, into *informal search*, users have now chosen a specific topic to focus on and concentrate on attaining as much information about that topic as possible. Notice the shift from browsing to *extracting*, now users have stopped viewing random web pages, and have started to perform focused searches on a specific key topic, using the information gained during the search process to define more precise search queries. Also notice that users have finished differentiating between relevant and non-relevant information, and have now concentrated on revisiting bookmarks of the pages related to the selected topic. The *formal search* is the final stage of the model, where users have attained enough knowledge to perform an accurate and efficient search which will return the web pages they need to fulfill their information need.

Belkin [8] proposed the theory that a search session is not comprised of a single search strategy, but instead is comprised of several different strategies, applied at different stages of the search process. Belkin suggests that the change in strategies during the search process is a result of the changes in user's knowledge and goals over the course of a single information-seeking episode. This model was built around the concept of a multi-dimensional space from which a multitude of different *Information Seeking Strategies* (ISS) can be described. Table 2.5 contains a list of the ISS and their associated variables; there are four facets, and two

	method		goal		mode		resource	
ISS	**scan**	**search**	**learn**	**select**	**recognize**	**specify**	**info**	**metadata**
01	x		x		x		x	
02	x		x		x			x
03	x		x			x	x	
04	x		x			x		x
05	x			x	x		x	
06	x			x	x			x
07	x			x		x	x	
08	x			x		x		x
09		x	x		x		x	
10		x	x		x			x
11		x	x			x	x	
12		x	x			x		x
13		x		x	x		x	
14		x		x	x			x
15		x		x		x	x	
16		x		x		x		x

Table 2.5: **Belkin's Information Seeking Strategies (ISS) [10].** Belkin theorized that during a search session, users would execute several different search strategies, as the search progressed and their information need changed. Above is a table of the different ISS strategies, each row represents a different ISS, and each column represents its characteristics. Note, the ISS names have been abbreviated in order to fit them in the space above.

dimensions associated with each facet. The four main facets, the *method* of interaction, the *goal* of the interaction, the *mode* of retrieval, and the type of *resource*, have all been identified through observation and classification of information-seeking behaviours in a variety of settings (see [8]). These are further subdivided into eight dimensions, and each ISS is defined by a unique combination of these dimensions.

Looking at the dimensions of ISS in more detail, searching and scanning refer to the *method* by which users begin their search. In *searching*, it is implied that users already know which specific area to look in (note this differs from previous usage of the term 'searching' in the thesis, which is more general in context). *Scanning* implies that users have to search over a large area, not knowing where to start their search. In terms of web-searching, scanning would imply that users have to search a list of search results or directory listing before deciding which website to begin looking at, whereas searching would refer to users searching the different web pages in a specific website, or searching for a piece of text or keyword within a web page. The *goal for interaction* is influenced greatly by the starting knowledge of users; if they have a clear idea of their objectives, then the task is simply a job of *selecting* the correct information and reviewing it, or storing it. If the objective is vague, then they must explore and discover information through a process of *learning* that may help define their objective. The *mode* of the search, refers to the type of search conducted, with *specification* referring to a search with a known purpose (similar to the directed search), and *recognition* referring to a undirected search (similar

to browsing). The type of *resource*, can either be *information* which is directly related to the search (text, audio, images), or can be *meta-information* which can be useful when attempting to filter your results (e.g. file size, number of links).

	method		**goal**		**mode**		**resource**	
ISS	**scan**	**search**	**learn**	**select**	**recognize**	**specify**	**info**	**metadata**
2	x		x		x			x
15		x		x		x	x	

Table 2.6: **Examples of two ISS from [10]:** ISS 2 bears similarity to the start of a search, where users are browsing without a clear objective, generally scanning the presented data. In contrast, ISS 15 bears more similarity to the end of a search, where users have identified what specific information they need to fulfill their information need, and are currently searching the web page. These strategies are explained in more detail below.

The structure of an ISS is best illustrated with an example, seen in Table 2.6. *ISS2* represents a situation in which users are dealing with an unspecified and unformulated problem which they must *learn* more about. They may perform a web search, *scan* the *meta data* returned, and attempt to *recognize* salient words in the search results returned. This may represent the beginning of a search. By contrast *ISS15* could represent the end of a search, where users have *selected* the webpage to view, know *specifically* the piece of textual *information* they are looking for, and are currently *searching* the webpage for the information. In this section, several hybrid models have been shown. Hybrid models demonstrate how unstructured search models have become more strictly defined, with specific search actions being associated with specific strategies. However, as the strategies became linked to specific search actions, the strategies begin forming more rigidly defined *structured search models*.

2.2.3 Structured search models

A search session can be defined by the actions that are performed during the search; these *search actions* can be ordered in such a fashion that they play out a pattern of common behaviours performed by users. For example the actions, 'submit query', 'read search results' and 'open web pages' can be placed in order, to form the start of a search. Linking these patterns together into a structure can be used to form a *search pattern*, and the linking of several search patterns into a diagram forms a *structured search model*. Unlike the previously mentioned strategies, these define the low-level actions of the user, giving a more detailed view of the search process. Both the *search actions* and the *search patterns* are building blocks of the structured search models. Tauscher et al. [114] performed a study of student's revisitation patterns when searching the World Wide Web. As a result, Tauscher et al. drew up a list of several search patterns which occur during the student's searches which were explicitly based on the combinations of different

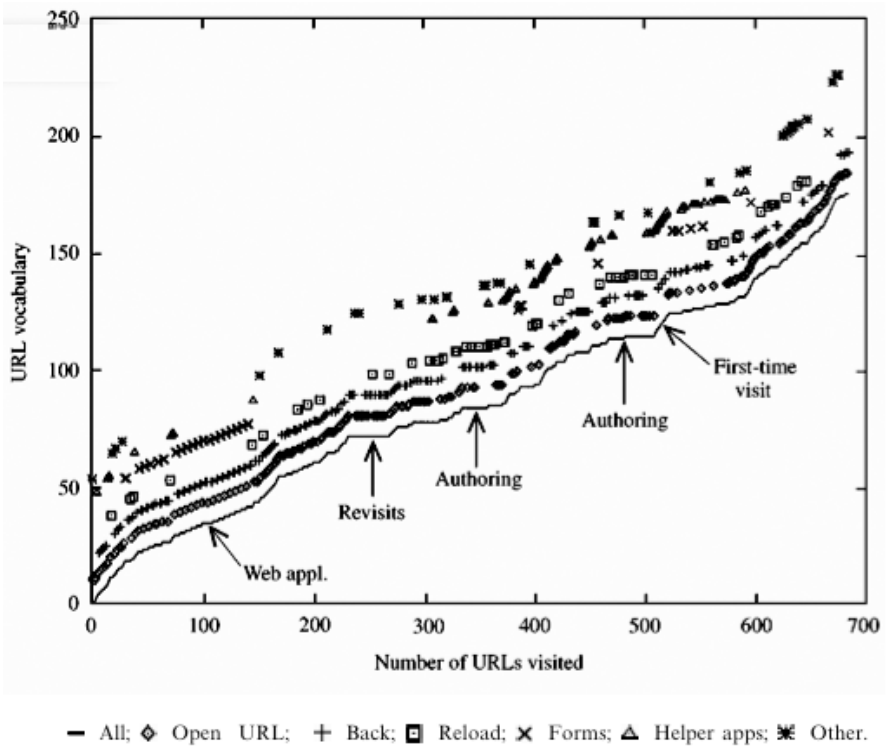

Figure 2.6: **Revisitation patterns: a student's search session (from [114]:** This data is from Tauscher et al's [114] study on revisitation patterns. This shows the data retrieved from a student, as the search progressed. Progress can be tracked along the x-axis by following the number of URLs visited. The y-axis tracks the URL vocabulary, the number of different web pages that were visited (e.g. when a page is revisited, then this does not increase). The icons display the different tools that the student used at that particular point in the search. By analyzing a combination of the tools used, URLs visited, and changes in the URL vocabulary, Tauscher was able to identify several different search strategies in play during the search.

search actions taken by users during the search.

Looking at Figure 2.6 an example of the type of data that Tauscher et al. collected for the study can be seen. Subject progress was measured in number of URLs visited, and changes in URL vocabulary (number of new pages visited), which were used for the x and y axes of the graph respectively. The bottom-most diagonal line represents subject progress, whereas all the other lines of symbols represent the different search functions used at each stage of the search (e.g. Open URL, Back button, Reload). Tauscher et al. identify and point out several different patterns in play based on a combination of the functions used, URLs visited, and changes in the URL vocabulary, at each point in the search. These formed the basis of Tauscher's seven Search patterns (below).

- **First time visits:** This happens when users first comes across a cluster of new pages. This appears in the subjects data as a steeply sloped area (see figure 2.6) because the users begin viewing a large number of new pages (increasing the URL vocabulary).

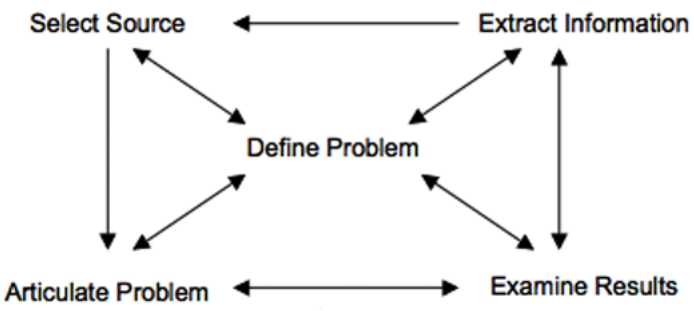

Figure 2.7: **A basic model of information-seeking by Marchionini et al. [73].** This shows the basic actions users perform during an information-seeking episode. The searcher starts in the middle, where the problem is defined (information need), then articulates the problem (translates the information need into query terms), examines the results, extracts relevant information and then if the information need is not satisfied uses the newfound information to select a new source of information and once again begins articulating his problem. Note that reformulation and changes to the information need are implied, by the arrows pointing from the various search actions back to the central information need.

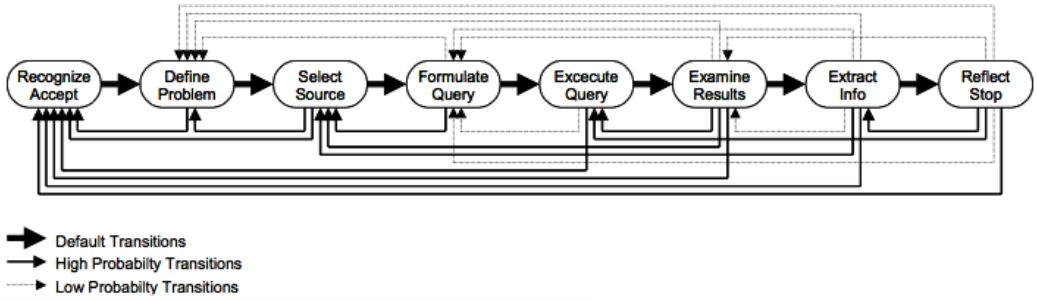

Figure 2.8: **Marchionini's second model of information-seeking (as seen in [71]).** Unlike in Marchionini's first model [73], this has a more definitive start and finish. The details of interactions in this model are shown at a higher level of detail, to the extent that even the probability of transitions taking place is calculated and displayed as different types of arrows on the model.

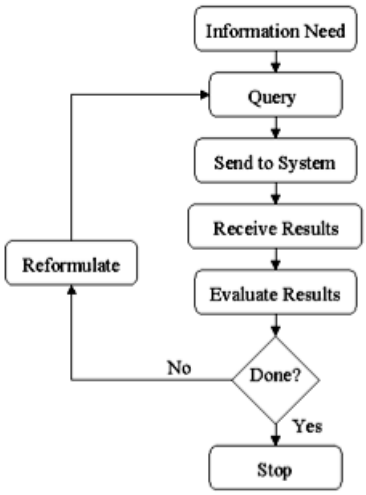

Figure 2.9: **Hearst's model of information access processes (from [43]).** This shows another basic model of information-seeking. Unlike in Marchionini's model [73], reformulation has been stated explicitly as a search action.

- **Revisits to pages:** When users revisit web pages. Often distinguished by plateau areas (showing no change to the url vocabulary) and several back to back occurrences of 'back button' usage.

- **Authoring of pages:** This involves a tactic unrelated to web searching, which is the authoring of web pages, where certain web pages are refreshed multiple times in order to review changes made to that page.

- **Regular use of web-based applications:** This happens when applets, web forms or other web applications are used. This is displayed on the subject's data as a moderately sloped area with a combination of 'open URL', 'back button' and 'forms activity'.

- **Guided tour:** Where users visit a web page which include a set of structured links (such as a tutorial, consisting of several web pages, where a 'next' button is used to move onto the next page). This is characteristically displayed on the subjects data as consecutive URL open actions.

- **Breadth-first search:** Also known as *Hub and spoke* searching, interaction takes place around a central web page (hub) from which users navigate to links to many other web pages. This can be seen in the subject data as alternating occurrences of open URL and back actions. These actions are consistently used to return to an index page to access other URLs on the page. Note, this behaviour is also apparent when presented with a list of search results, to which users consistently return.

- **Depth-first search:** Where users follow links deeply before returning to a central web page. This is represented in the data by a series of open URL actions followed by several consecutive back actions.

It is interesting to note that some of the search patterns (such as *breadth-first search*, and *depth-first search*) can be seen as more accurately defined components of previously discussed unstructured search models. Take for example, a scenario where users perform a search, then browse the results, viewing web pages and then jumping back to the search result list (breadth-first search). Occasionally they will find an interesting web page and may attempt to learn more about the topic presented by the web page by continually following links from page to page (depth-first search). However search patterns are still merely search actions with some added linear structuring. True structural search models have a clearly defined start and finish to the search, as well as descriptions that connect the various search actions into a cohesive structure, that don't necessarily operate linearly.

Marchionini' s [73] model is an example of a basic structured search model for information-seeking (see figure 2.7). Several of the standard search actions are defined and linked into a

structure, starting in the centre with the information need, and moving around the centre of the structured diagram. Notice that in Marchionini's model, the paths followed in the search are not always linear, and arrows connect to the centre from several parts of the search, allowing users to recurse and retrace over paths they have previously visited. The implied recursion is evident in the arrows pointing from various search actions, back towards the centre, mirroring the changing of the information need as the search progresses, and they gather more information. This will often lead to users reformulating their query and generating new searches. A second, more detailed version of Marchionini's model was later developed (see Figure 2.8), which defined the start and finish of the search explicitly as well as providing more information on the transitions between the actions. Note that, unlike Marchionini [73], Hearst [43] adds reformulation as an explicit search action within her model of information access processes (see figure 2.9).

Not all structured search models operate at the same level of granularity, because different authors tend to concentrate on emphasizing different aspects of the search process. A common example, can be described as the *macro-model* or model of the gross information-seeking behaviour [122]. Like other structured search models, these display the relationships between the different search processes explicitly, however they tend to operate on a higher level, and often include the information resources and systems being accessed, as well as the affecting psychological and environmental factors. Ingwersen [53] built an expansive model that added the effect of environmental factors upon a user's search process (see Figure 2.10). Ingwersen was concerned with identifying processes of cognition which may occur in all the information processing elements involved. This model shares some basic interaction processes with most other interaction models, but expands to include the *user's cognitive space*, and users external *environment*. The right-hand side of the figure shows the individual user, as well as all the factors affecting him, such as the user's existing knowledge structures, goals and desires. The *social environment* affects user's inbuilt systems of categories and concepts, and will also affect their information preferences, searching conventions and cognitive structures. Notice also, that both the system and the information objects handled by users are explicitly detailed in the model. A similar cognitive model to Ingwersen's is Saracevic's [97] *Stratified Interaction model* (see Figure 2.11). This too specifies the physical resources used, and the effects of the external environment upon the search behaviour of users. Wilson [121] posited the theory that because Macro-models and Structured search models operated at different levels of abstraction, they could in effect be 'nested' together, to form a more descriptive view of the information-seeking process. The example used was 'nesting', Ellis's [30] search behaviour models (see section 2.2.1) within his own model (see Figure 2.12).

This section has illustrated the fact that there are several different models of information-seeking, most of which share common elements, but each of which places emphasis on different

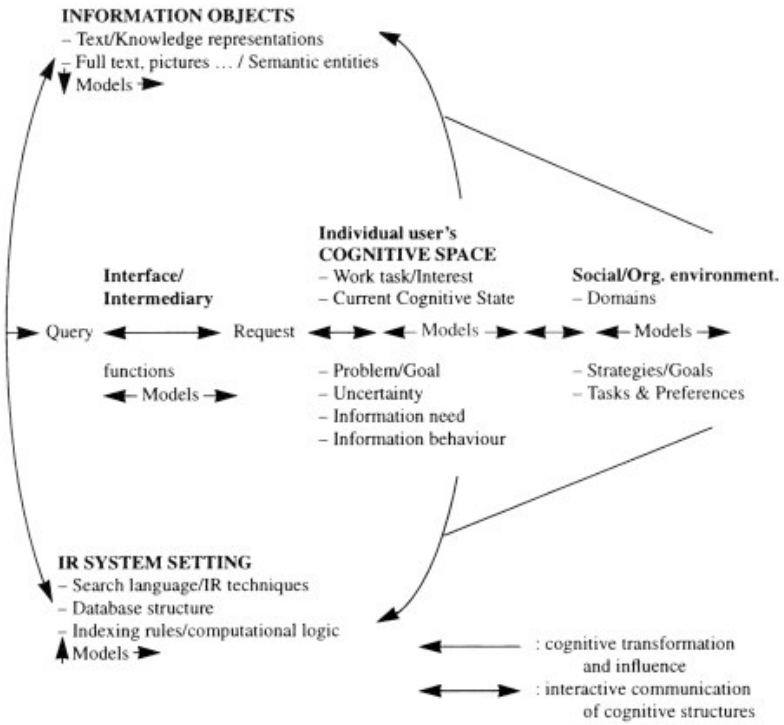

Figure 2.10: **Ingwersen's model of the IR Process (from [122]).** This model of information-seeking added the influence of external factors, as seen in the diagram as *cognitive space* and *social and organizational environment*.

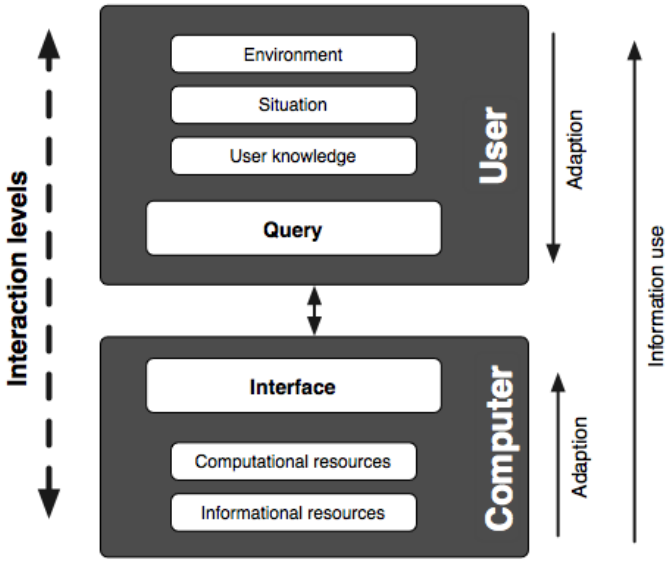

Figure 2.11: **Saracevic's stratified model of the IR Process.** The model is comprised of objects interacting at different levels (strata). At the top-end, users (represented by their own knowledge, goals etc.) are affected by a situation (task, problems) as well as the environment. The interface with which users interact, relies on computational resources to calculate the best fit to the query, as well as the informational resources, (text, images, and meta information) needed for representation.

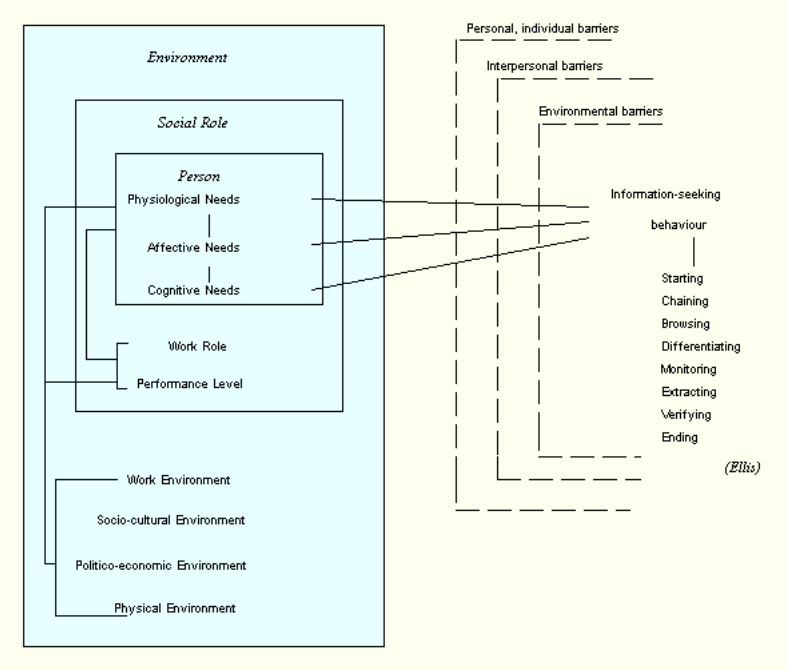

Figure 2.12: **Wilson's 'nested' model of search behaviour (from [122])** Wilson combined his higher level Macro model [121] with Ellis's Search behaviours [31] to form a more detailed model of the search behaviour of users. Ellis's search behaviours are 'nested' within the person's needs, and are affected by various factors in the user's environment.

information-seeking tasks. The next thing to consider is where the evolving search theory and the berry-picking model fit in the information-seeking process and how they relate to other information-seeking models.

2.2.4 Discussion

The evolving search theory and berry-picking model are aspects of information-seeking behaviours that take place when users begin a search with little or no context information, or with only a vague idea of what they are searching for. In a way evolving search could be considered a browsing strategy, but explained in more detail. The influences of evolving search can be seen in nearly all the models presented. This can be demonstrated by looking at the key features of the evolving search and berry-picking model.

In an evolving search, users begin with an information need which is not clearly defined. It may be that users begin the search with very little information or are uncertain as to what they are searching for. Either way, users will find it hard to translate their information needs into an efficient set of search terms. This feature is a common feature of all browsing strategies, and encompasses all three types of browsing demonstrated at the start of the chapter (browsing because of *limited information*, *unfamiliar territory* and *unclear objectives*). Since the browsing strategy is a common feature of nearly every model, it is not surprising that the evolving search

incorporates it as part of its theory.

In an evolving search, users will generate multiple searches based on what little information they have. This is because of an inability to generate an efficient initial search, which forces users to follow clues and harvest information that may lead them closer to fulfilling their goals. This feature is more clearly expressed in the structured search models. Both Marchionini [74] and Hearst [43] show users reformulating their queries (and hence generating multiple searches) as part of their models. Spink's [103] elements of the interactive search process describe the search process as a cycle of interactive feedback loops, where multiple searches are formed, and information gathered from previous searches is cycled into the generation of future searches.

In an evolving search, information needs change over the course of the search. As stated in the evolving search model, as the users are exposed to different information sources, their needs will change, either becoming more focused, or branching off into other areas. This feature is clearly described in structured search models, where the reformulation stage is shown to sometimes feedback into the users information need. Note that some models, such as Hearst's [43] model, assume that the information need is unchanged throughout the search session.

User's strategies change over the course of an evolving search. Because users begin with an ill defined information need, they will initiate a very broad search, which changes as the search progresses. As information needs change, so will the types and sources of information that are needed, requiring users to adopt different strategies. This feature is best expressed by the hybrid search models. The models of both Ellis et al. [31], and Belkin [9] suggest that a search model is comprised of several different strategies that can take place in any order. To this end, each model is presented as a set of strategies that can be combined in different orders to describe a search session.

In an evolving search, an information need is satisfied by multiple sources. As stated in the berry-picking model, users performing evolving searches will gather information from multiple sources, all of which will contribute to answering the information need. As a result the information need is not solved by any single set of retrieved results. This feature is the only one that is not expressed explicitly by any other model. It can be argued that this feature is implied in models which show users performing multiple searches, but it is never stated.

The evolving search shares many features with other models described, but are also different in other features. Many of the higher level models do not explicitly state the possibility of users generating multiple searches. Many more models (e.g. Hearst [43], Belkin [9], Choo [20], Shneiderman [100]) make no mention of the fact that the information need can change during a search session. Most of the hybrid and unstructured search models allow the flexibility of changing strategies through out the search, although this happens less often in structured search models. No mention is made in any of the models about how the information need is satisfied;

it is naturally assumed that users will eventually form a query that will provide the answer to their question, whereas in reality information needs are satisfied by information drawn from different sources. Later chapters will discuss the effects generated by these different features of the evolving search model (e.g. multiple searches generate a large amount of data), and discuss how each of these effects cant be addressed.

2.3 Studies of Search behaviour

Looking at theoretical models of information-seeking helps build a knowledge base of the search process, however, to truly understand users, experiments must be performed to capture their search behaviour. Much research has been dedicated to examining the users of web search engines (Cacheda and Vina [17], Holscher and Strube [50], Jansen and Pooch [54], Jansen and Spink [56], and Montgomery and Faloutsos [78]). This section presents a summary of the major trends present in the data, and a more detailed view of these results can be found in Table 2.7.

The query data showed that an average a search session consists of either two or three queries. Silverstein et al. [102] reported an average of 2.02 queries per session, Jansen and Spink [55] reported an average of 2.8 queries per session, Wolfram et al. [125] reported 2.5 queries per session. The average query length is around two keywords. The range for the query lengths was surprisingly small, on average lying between 2.3 and 2.4 keywords per query for the studies of Silverstein [102], Jansen [55] and Wolfram [125]. One interesting pattern demonstrated that even the most frequently occurring query terms represent a small percentage of overall term usage. In Silverstein et al. [102] the most frequently used term (the word *'free'*) accounted for only approximately 0.6% of all term usage. This shows that users are now accessing a broader range of topics. A look at the term co-occurence in the result set showed that several two term phrases from different topics (e.g. art, news events, countries and entertainment) were among the most popular pairs reinforcing the theory of topic diversity.

Looking at search behaviour, the majority of studies indicate that users very rarely go beyond the first page of search results returned. If their information need remains unsatisfied, they either reformulate their query or perform another search. Silverstein et al. [102] reported that 85.2% of searchers did not view more than the first page of results, Jansen et al. [58] reported 72.8%, and Jansen and Spink [55] reported 83% and 76% for their AllTheWeb.com studies done in 2001 and 2002 respectively. However a pair of older studies of users of the Excite search engine, by Wolfram et al. [125], indicate a different trend. Their 1997 Excite study shows that only 28.6% of users viewed just the first page of results, with a more substantial 51.9% viewing three or more pages of results. The 1999 Excite study seems to be moving more in the direction of the main trend, with 42.7% viewing only one page of results. However it is nowhere near the 70 and

		AltaVista 1998	AltaVista 2002	AllTheWeb 2001	AllTheWeb 2002	Excite 1997	Excite 1999
Mean terms per query		2.35	2.92	2.4	2.3	2.4	2.4
Terms per query:	*1 term*	25.8%	20.4%	25.0%	33.0%	26.3%	29.8%
	2 terms	26.0%	30.8%	36.0%	33.0%	31.5%	33.8%
	3+ terms	27.6%	48.5%	39.0%	34.0%	43.1%	36.4%
Mean queries per user		2.02	2.91	3.0	2.8	2.5	1.9
Users modifying queries		20.4%	52.4%	47.0%	41.0%	52.0%	39.6%
Session length:	*1 query*	77.6%	47.6%	53.0%	59.0%	48.4%	60.4%
	2 queries	13.5%	20.4%	18.0%	16.0%	20.8%	19.8%
	3+ queries	6.9%	32.0%	29.0%	25.0%	30.8%	19.8%
Pages viewed:	*1 page*	85.2%	72.8%	83.0%	76.0%	28.6%	42.7%
	2 pages	7.5%	13.0%	10.0%	13.0%	19.5%	21.2%
	3+ pages	7.3%	14.1%	7.0%	11.0%	51.9%	36.1%
Use of modifiers		20.4%	20.0%	1.0%	1.0%	5.0%	8.0%

Table 2.7: **Data from Search behaviour studies.** This table details Search behaviour data for users of three search engines (AltaVista, AllTheWeb and Excite), in six separate studies. Both the 1998 and 2002 AltaVista data were collated by Jansen et al. [58], although the 1998 data was based on the work of Silverstein et al. [102]. Both of the AllTheWeb data sets were reported by Jansen and Spink [55], and the Excite data was collected by Wolfram et al. [125]. All of this data was collected from automated logs of user's search sessions.

		AltaVista	AllTheWeb	Excite
Mean terms per query		+0.57	-0.1	0
Terms per query:	*1 term*	-5.4%	+8.0%	+3.5%
	2 terms	+4.8%	-3.0%	+2.3%
	3+ terms	+20.9%	-5.0%	-6.7%
Mean queries per user		+0.89	-0.2	-0.6
Users modifying queries		+32.4%	-6.0%	-12.4%
Session length:	*1 query*	-30.0%	+6.0%	+12.0%
	2 queries	+6.9%	-2.0%	-1.0%
	3+ queries	+25.1%	-4.0%	-11.0%
Results pages viewed:	*1 page*	-12.4%	-7.0%	+14.1%
	2 pages	+5.5%	+3.0%	+1.7%
	3+ pages	+6.8%	+4.0%	-15.8%
Use of Boolean queries and modifiers		-0.4%	0.0%	+3.0%

Table 2.8: **A Comparison of the changes in Search behaviour.** Each of the three search engines mentioned in Table 2.7 are compared, so as to extract any changes or trends in users search behaviour. The results are displayed in the three columns to the right, where values representing the changes are given (either positive or negative).

80 percent figures quoted previously (see Table 2.7 for a look at Wolfram's data). A possible explanation may be that both of Wolfram's studies were conducted on users of the search engine Excite, whereas Silverstein and Jansen et al. ([58]) utilized the AltaVista search engine, and Jansen and Spink utilized the AllTheWeb search engine, and the difference in trends merely reflects the difference in user's search behaviours with different search engines. This will be discussed in more detail later on.

The results by Jansen et al. [58] showed some interesting trends. It was reported that the mean session duration was around an hour with a standard deviation of around 3 and a half hours. However Jansen believed that these results were skewed, due to several abnormally longer sessions. 81% of searches were less than 15 minutes, and a near 72% were less than 5 minutes long. Combined with a greater session length, this seems to indicate that users are performing more searches, but in a smaller amount of time (implying more scanning and less viewing of results).

It was was also noted that the majority of searchers did not utilize boolean operators or modifiers. The highest number of users who used boolean operators in their searches was 20%, in both the Silverstein [102] and Jansen et al. studies [58]. Both the 2001 and 2002 AllTheWeb studies [55] quoted a mere 1% of users, and Wolfram [125] reported less than 10%. Note that, although sexual topics are the most popular search topics, they only share a small percentage of the total number of searches with many other topics. It was also noticed in a temporal comparison of two search studies (the Jansen et al. [58] study), that search topics are greatly influenced by current trends.

Jansen and Pooch [54] reviewed web searching literature and compared web searchers with searchers of traditional web-searching literature. They reported that web searchers exhibit different search characteristics than searchers of other information systems. However drawing any definitive conclusions from the data can be difficult, because the studies are not unified in their approaches. Additional complications arise because most of the data is accumulated from automatic logs of user's searches, which can often lead to ambiguities about the accuracy of the data. The problems with comparing data from these search result studies are presented.

Different studies have different definitions for what constitutes a query and what constitutes a request. In general it is understood that a request consists of an interaction made with the search engine to elicit information; it can be a standard search query, a null query (no terms) or a request for additional results (pressing the 'next 10 results' button). A query is composed of the terms entered into the search engine, as well as the action of submitting the query to the search engine, and is a specific type of request. The problem is that some of the studies interpret requests for additional results, and null queries as actual queries. This can be problematic when attempting to calculate the number of queries that users perform or the number of results users

view in a search session.

Related to the problem of queries and requests, is the problem of identical queries. For certain studies, requesting the next page of results is recorded as performing the same identical query a second time. This makes it hard to determine when users are viewing the next set of results, recalling the page of results after visiting another page (via the back button), or simply making a mistake and re-performing the search with identical search terms.

Search logs also differ as to what constitutes a search term, which in turn affects the recorded number of queries per turn. Normally one would expect a search term to refer to a unbroken string of characters, however Silverstein et al. [102] treat words within quotation marks (used when denoting an unbroken string in a query) as a single term. For example the query ["africa safari botswana"] would be considered to be a single term query in Silverstein et al's study [102], but would be a three term query in Jansen et al's [57] study. On the other hand, Silverstein et al. [102] would treat queries like [host:www.safari.com] as a four term query (host, www, safari and com) whereas Jansen et al. [57] would consider it a single term query.

It is difficult to determine how many search results users view because of the way the search log data is collected. Similarly following users actions after they have left the search engine is impossible when only using search logs. Most of the studies instead reported the number of pages of search results visited. However even comparing these can be difficult, for example, Silverstein et al. [102] record the number of results screens viewed per query, whereas Jansen et al. [57] record the number of results screens viewed per user.

These problems causes inaccuracies which are detrimental to observing evolving search behaviour. Web page browsing behaviour is an important part of the evolving search and not being able to monitor this data is problematic. Similarly not being able to differentiate between identical queries and requests for more search results hampers any analysis on users recalling and re-performing of searches.

2.3.1 Changes in search behaviour

There is some argument over the change in search behaviour over the years, with varied and conflicting pieces of data reported. A comparison of the changes over time (based on the data in Table 2.7) can be seen in Table 2.8. Each pair of studies for each search engine is compared, and is represented by a column on the right of Table 2.8. The values in the columns represent the changes in search behaviour (either positive or negative) over time. A four year analysis of the Excite search engine, by Spink et al [104], showed that web searching sessions and query lengths have remained relatively stable over time, although a shift was noticed from entertainment to commercial searching. Jansen et al. [58] performed a study comparing Altavista.com search results (the 1998 data, was based on Silverstein et al's [102] study) temporally, with results from

4 years before. Jansen and Spink [55] performed a similar comparison based on studies of users of the AllTheWeb.com search engine, in both 2001 and 2002. Wolfram et al. [125] reported two sets of results from previous studies on users of the Excite search engine, in 1997 and 1999. Key trends were observed related to users queries, sessions, results pages, and syntax.

Results detailing query lengths varied, data from the AltaVista study seemed to indicate a drop in one term queries, and an increase in three or more term queries, which may indicate a move towards more finely tuned searches, and more specific query formulation. However, data from the AllTheWeb studies show the opposite, a small increase in one term queries and decreases in two and three term queries. The Excite study showed yet another variation, with small increases in one and two term queries, and decreases in 3 or more term queries. AltaVista reported an increase in users modifying their queries, quoting a 32.4% increase in modified queries, between 1998 and 2002. The results showed a move towards greater interactivity between users and the search engine, characterized by an increase in query modifications. However the AllTheWeb and Excite studies show drops of 6.0% and 12.4% in query modifications respectively. This may signify an increase in search engine accuracy, and being able to return relevant results more often.

Looking at session length, the AltaVista data seemed to suggest a drop in users performing single queries, and a move towards producing three or more queries (increasing by 25.1% between 1998 and 2002). However both the AllTheWeb and Excite studies show a contrary trend, with a decrease in users generating two and three or more queries and an increase in users generating single queries. Unsurprisingly the number of results pages viewed also varies. This time, the AltaVista and AllTheWeb studies share common ground, with decreases in users only viewing one page of results, and increases in viewing two or three pages of results. This may indicate a move towards more persistent search, or less efficient search algorithms that fail to return relevant results in the first 10 results. The Excite results seem to show a contrary move, with a sharp increase in users who only view a single page of results and a steep drop in the users viewing more than one page of results. Note that there was little or no change to the amount of Boolean operators and modifiers used. The Altavista and AllTheWeb data showed differences of less than 1% and the Excite data only showed a 3% shift.

By investigating the changes in user's search behaviour over time several interesting trends were revealed. The biggest surprise was the numerous conflicting results presented by studies which shared similar authors (Bernard J. Jansen and Amanda Spink contributed to most of the studies discussed). A possible explanation is that the different trends in changes in search behaviour can only be attributed to the differences in search engines and how they affect the user's search behaviour.

Another affecting factor may involve the amount of time spent conducting the studies, as

well as the amount of time in between studies. The majority of studies spanned 24 hours or less, thus the data acquired represented only a small slice of search behaviour compared to say a month, or a year's worth of data. Search behaviour of users may vary depending on the day of the week, for example users may be inclined to search for more work-related web pages during a weekday, or more entertainment related web pages close to holidays. The time-span of the comparisons are also different; the AltaVista study compares studies with a four year difference, whereas both the AllTheWeb and Excite studies only have a two year difference in the studies compared. Closer inspection of the data shows that while the different data sets show conflicting changes in search behaviour, they do little to contradict previous theories on user's search behaviour.

The previously stated theory about the number of search results pages viewed was that users only view a single page of results before either finishing their search or reformulating their query. While some of the search studies show increasing trends in users viewing more than one page of results (see Table 2.8), it can be seen that these increases are superficial (none greater than 7%) and if looking at Table 2.7 it can be seen that with the exception of the 1998 Excite data, all the other studies show a majority in users only viewing one page of results, four studies quoting over 70% of their users displaying such behaviour.

Also previously stated, a theory significant to the evolving search is that users aren't always satisfied with the first query they perform. Many users perform directed searches and will get their results with their first query, but it is assumed that at least as many users will be unable to define an efficient query and will need to perform a second or third search. Some of the search trends show a decrease in the number of users performing more than one search, but these differences are marginal at best, the majority of decreases being less than 5% (the largest decrease being a mere 11%). This is reflected in the mean queries per user, where both the AllTheWeb and Excite studies show changes of less than 1%. With the exception of the 1998 AltaVista results, all of studies report at least 40% or more users performing more than one search, which fits with the current theory of multiple searches and browsing strategies.

In summary, the results presented here fit with current views of web searching. The evidence shows that users often perform more than one search, which supports the multiple search aspect of the evolving search. One criticism of observing user's search behaviour through search engine logs is that you cannot observe their actions outside of the search engine (e.g. writing down, comparing and recalling information). These are all essential parts of the evolving search theory, and need to be captured in order to further support research in the area.

2.4 Overview

In this chapter the models and theories of information-seeking have been presented and discussed. The two primary strategies of directed searching and browsing have been explained, and the distinction between information-retrieval and information-seeking has also been made. Information-seeking models can be subdivided according to the types, unstructured search models, hybrid search models and structured search models. Each of these was explained with examples. The qualities of the evolving search as an information-seeking model were discussed and compared to existing models, and observations were made as to the similarities and differences.

It was noted that the evolving search shares common ground with the majority of presented information-seeking models. Very often users start with an unclear information need, and multiple searches are often generated. However, the evolving search contains aspects of search behaviour that are only present in some of the models presented. This includes the possibility of information needs changing during the search, and the changing of user's search strategies during the search. One aspect of the evolving search, that was not present in other models was the fact that information need is satisfied by information gathered through out the search. The next chapter will discuss the technological aspects of web searching, as well as describe the visualization techniques available to tackle the problems presented by evolving searches.

Chapter 3

Visualization, Techniques and Technologies

Designing an effective evolving search tool is a challenge, and one must carefully consider the combination of different display techniques, interaction techniques and technologies that will be utilized. When choosing the right *visualization display technique*, developers need to work out what information is most pertinent to users, and how to most effectively lay out and display that information. Choosing a set of suitable *interaction techniques* that allow users to interface with and manipulate the data is also important. In addition to the standard set of search result interaction techniques (buttons, menus), other more novel techniques such as the distortion of data (using focus-and-context technologies), pop-ups and highlighting, can all be used to help users in their search interactions, whether they be browsing, comparing or recalling data. The *technology* used will also affect the visualization, not only in what techniques can be utilized, but also in what data is available for visualizing. For example, in addition to the standard search result information (title, summary, url), different search technologies also make available different meta-data (external/internal links, page size) and statistical data (e.g. image count, key word count, visitor count) related to the web pages and web sites returned from searches.

This chapter provides the related work and concepts that underpin the visualization designs discussed later in this thesis. The chapter presents different visualizations, techniques and technologies designed by current researchers, that can aid users in performing evolving searches. The chapter is divided into three main parts. The first section looks at the two principal components of *web searching*: the search engine and the web browser, and explains the underlying processes behind web searching as well as discussing the tools currently being used to search for information on the WWW. The second section discusses the various *evolving search interface issues* outlined by Bates [6] (see chapter 1 - section 1.3) and explores the different visualization

techniques that could be used to tackle these problems. These are the issues of *information visualization*, *information recall* and *opportunistic searching*. The third section looks at two *online interface issues*, which were not included in Bates's list of interface issues. These were not apparent at the time of Bates's research because they are specific to the online environment. These are the problems of *information-seeking*, and *information management*.

Figure 3.13: **A Map of the Internet [76] (from http://www.ethicalmedia.com/).** The internet is a vast collection of online resources, residing over thousands of servers, connected by millions of hyperlinks. Finding information on the internet without the aid of a search engine can be difficult.

3.1 Web searching

There is no easily navigable map of the Internet (see Figure 3.13); getting from website A to website B might be as simple as typing in a website address, but discovering the website address first can sometimes be difficult. There are different ways for users to find websites of interest, for example, some users consult sources external to the WWW (via posters, radio, or word of mouth). This is generally considered the least effective method, because users have to remember website addresses, as well as recall the addresses the next time they are at a computer. Other users might discover information from sources within the WWW, for example, when browsing known web pages, users will often come across links and web advertising which will lead them to new websites. However, this too is limited to what web pages users already know and the pages linked from them. The easiest method of finding information on the WWW is to use a search engine, because users only need to remember one (search engine) website address to gain access to a thousand others. Each search engine is different in the web pages it indexes, the conventions it uses, and the tools it provides, however, most search engines provide the same

basic functionality. This section will present the two fundamental components needed for web searching, the *search engine* and the *web browser*.

3.1.1 Search engine

All web search engines consist of the same basic elements, a text box to enter your query and a submit button (see Figure 3.14). Finding information with a search engine is a deceptively simple process: users only need to type in the words that best describe the information they are seeking and press submit. All that is required is to type words which best describe the search and press submit. At that point the experience will diverge based on the search engine being used.

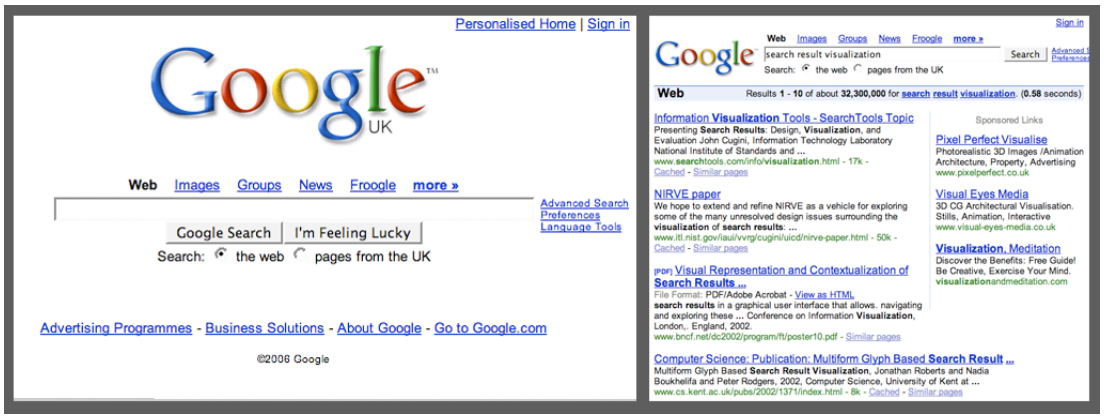

Figure 3.14: **The Google search engine [36]**. This is a view of a standard web search engine interface. On the left is the search engine's front page, which has a text box for entering your query and a button to submit your search. On the right is the list of search results returned from the search.

The majority of web search engines will return a textual list (normally displaying 10 results per page) of ranked document surrogates (ranked according to the search engine's own algorithm), which are referred to as *search results*. Each search result consists of a page title, a snippet of text from the web page, and a URL address. Clicking on the title will often take users to the web page directly and various types of meta-data (page size, file type) are displayed as well as links to more advanced functions ('find similar to', 'open cached page'). Many search engines such as Altavista.com, Yahoo.com and Google.com utilize this form of interface, and in this work this type of search interface will be referred to as the 'traditional' search interface, because it is the most commonly used form of online search tool.

Most search engines provide additional content and interfaces in order to distinguish themselves from their rivals. Google.com offers web searching in a variety of different areas, including images, news, weblogs and maps. Clusty.com automatically categorizes search results into clusters for easy viewing. Groxis.com throws away the traditional interface, and instead groups

together similar web pages and displays them as hierarchies of circles. Some search engines known as *meta-search engines*, (e.g. dogpile.com) take information from multiple (other) search engines, and display the best combination of results.

Search engine technology

The core of a search engine is its web crawler (sometimes called a web spider), which is a program that automatically browses the World Wide Web in a methodical and automated manner. Web crawlers have a variety of functions, but they are commonly used for visiting web pages and copying them for indexing by search engines. Crawlers can also be used to provide maintenance for websites, checking code, and gathering specific types of information from web pages (e.g. harvesting email addresses for spam).

A web crawler begins by following every link it finds, analyzing the contents of the web pages found to determine what should be indexed (e.g. words extracted from titles, headings, meta tags). The data is then stored into an index database. When a query is performed, the search terms input are checked against this database, and a list of best matching web pages are returned. Different search engines have different ways of storing and recalling information.

The very first tool used for searching on the WWW was called 'Archie' (short for Archive), created by Alan Emtage. The program downloaded the directory listings of all the files located on public anonymous FTP (File transfer protocol) sites, and from this created a database populated with searchable filenames. This form of indexing was seen in later programs such as Gopher [75] and in turn programs were written to search these indices.

The first true web search engine was "Wandex", which introduced the use of web crawlers to create a searchable index (note that this was not the first *focused* web crawler [37]). Web crawlers very quickly became a standard for web pages. In 1994 the *Web Crawler* search engine was introduced, using technology that would allow users to search for any word in any web page. This system of searching was implemented in the search engines that followed (e.g. Lycos, Excite, Altavista, InfoSeek). At around the same time, sorted and ordered directories of web pages called *Web directories* were being introduced (primary of which was Yahoo! [127]). The two disadvantages of the system were that, (1) most pages had to be manually indexed to be placed into relevant categories, and (2) this method employs a searchable subject index so its search engine searches only the titles and descriptions of sites and doesn't search individual web pages. This meant that if the indexes are not updated regularly, then changes in website content will not be reflected in the index. This style of search was eventually integrated with web searching technologies.

Perhaps the most popular search engine to date is the Google search engine [36]. Google shares the majority of features of traditional search engines, and as such will be discussed as

an example of a standard web search interface. In 1998, Sergey Brin and Lawrence Page wrote about the *'Page Rank'* algorithm in their paper *'The anatomy of a Large-scale Hypertextual web search engine'* ([16]), and in that same year they used the Page Rank algorithm to develop the Google search engine. The Page Rank algorithm utilized link popularity in order to rank web pages, theorizing that good and interesting web pages, are linked to more often than others. Thus a web page's page-rank was (among other things) based upon which web pages linked to it, and the number of links on those web pages. It should be noted that each search engine uses different algorithms and methodologies to obtain search results, and the Page Rank algorithm is simply one example.

'Nielsen//NetRatings', a leading company in the area of market research, investigated the search behavior of more than a million people worldwide, through the use of real-time monitoring on computers. Their statistics [79] showed that out of the 5.1 billion searches performed in the month of November (2005), 46.3% of web surfers used Google.com, twice the size of their closest competitor Yahoo.com. The service has been so popular that the phrase 'to Google' is now used to describe web searching in general. Google continues to set the benchmark for search engines. In addition to the web search, Google also provides keyword searches of images and news sources. Less traditional search services have been introduced such as a map service, a weblog searching service, and shopping search service.

Search engine interaction

Google has been praised for a very clean interface, devoid of adverts and clutter, as well as a variety of interesting and unique services. For the more advanced users there are a multitude of other options available for web searching. This section will discuss various web search tools that are available to users to help manipulate their results, and show them results of interest (a topic that will be returned to, and contrasted in later chapters). There are two primary methods of refining a users search engine results, *boolean syntax* (the more commonly used, which varies little between different search engines), and *function syntax* (less commonly used, can vary widely between different search engines). Boolean syntax is used to include or exclude results from a search, based on a set of user specified criteria. By default Google uses the Boolean AND, but both Booleans (AND, OR) can be added directly into searches to modify them. The operators '+' and '-' can be used to automatically include and exclude words directly from a search.

Function syntax is used to restrict the search to a particular area. For example, Google provides commands to restrict the search to only the words in the title of the web page, the body of the web page, or the hyperlinks. The search can also be made website-specific, within a certain date-range, or for a specific file type. The search engine can also be indirectly manipulated in

more subtle ways, for example some search engines are affected by the use of capitalization, while other search engines (notably Google), are affected by the order that search terms are entered. Some search engines will remove *stop words* (common words such as 'the', 'and', 'or') from the search terms that a user inputs in order to improve the effectiveness of the search.

Search engines are not without their own unique set of flaws. There are ways to manipulate a search engine to display a particular website first ('Spoof your results') by including within the websites web pages, certain links and keywords. This can lead to more relevant search results being pushed below these fake results, lowering the chances that relevant results will appear in the first 10 results. Other flaws are inherently technical. This is related to the fact that the WWW is constantly expanding, and is growing much faster than current search engines could hope to index. Coupled with the fact that search engine web crawlers have to frequently revisit web pages to update them, it is very hard to keep search engines up-to-date. The textual format of the search query is also a problem. Because queries are limited to words only, there is always the chance of false positives being generated, for example, the phrase 'the big five' (among other things) refers to the five most dangerous safari animals, the five dominant psychological personality constructs and a chinese character encoding.

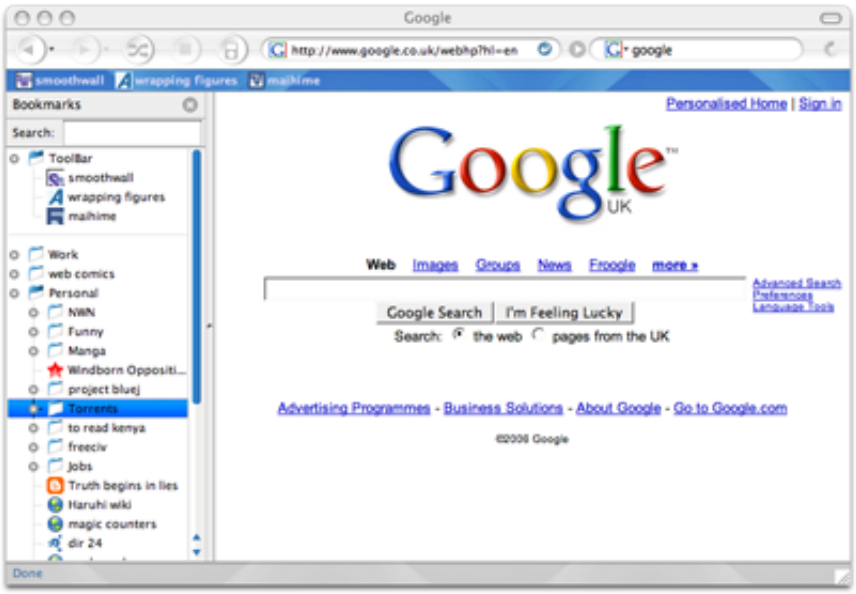

Figure 3.15: **Firefox web browser.** This shows a standard web browser. Along the top are the standard interaction buttons (refresh, stop, back, forwards , home) and the address bar. Along the right is a sidebar (which can be hidden away) which shows the web pages that users have bookmarked (the sidebar can also contain other functions such as the search history). At the centre/right is the browser's main view where the web page is rendered.

3.1.2 Web browser

Most web browsers consist of the same basic components, although most are customizable in terms of layout and components. Figure 3.15 shows a view of a standard web browser. Along the top is the web browsers toolbar from which users can interact with the web page using a standard set of interactions (*refresh*, *stop*, *backwards*, *forwards* and *home*). The address bar is also located at the top of the browser window, and sometimes doubles as a browser history (double-clicking on the bar will open the history). The web page is displayed in the main view below the toolbar, although some web browsers allow the use of tabs, allowing several web pages to be opened within a single browser window (e.g. Firefox [32] and Opera [98] web browsers).

The web browser contains different methods of recalling previously viewed information, and it is important to note that web browser history tools each have different continuity spans. Some exist on a (1) session by session basis (such as back-buttons and hyperlink highlighting), (2) are retained for a span of days (history lists), and (3) are permanent until deleted (bookmarking). Some web browsers allow users to save entire sessions of web pages, others such as the Opera [98] web browser automatically loads up the last pages you visited, each time you start the browser.

People use bookmarks in different ways. Some people use them as temporary flags, added opportunistically only to be discarded later, other people use them as landmarks which are stored and ordered carefully, so they can be recalled in the long-term. Consequently, the amount of bookmark usage can vary from person to person.

The back-button in the toolbar can also be used to recall information, a single click moving users to the most recently viewed page. Users can go back to previous pages through a pull-down menu which contains links to the last few web pages visited (usually between 5 and 10). Back-buttons are interesting in that they provide an automatic history system in which data on the user's progress are stored, and hence are an essential navigation tool. As a result they are used more often than other automatic history systems, but are limited in their functionality. Back-buttons act in a linear fashion (storing pages after they have visited them, in a linear list) and as a result are not well suited to browsing in a hyperlink rich environment, where users will often dart back and forth between pages. If a searcher should follow several links from a start page, return to the start page and then visit a different link, all record of the web pages that had been visited previously will be erased from the back-button's history. This problem is referred to in this work as the *branching problem*. Hyperlink coloring is an indirect approach to showing the browsing history of users. Textual hyperlinks are colored in a bright color and underlined to show that they are hyperlinks, but in addition to this, they change color when users have clicked upon them and visited their associated link.

The basic design of web browsers has not changed for nearly a decade. Many of the tools, have

been added to and modified slightly, but essentially all the functions remain the same. On the other hand, over the last decade, web searchers needs, search behaviours and the information available have all changed. Current web search tools do not provide suitable support for users during an evolving search. The following sections will present the interface issues addressed in this work, as well as discuss the different techniques that can be used to aid users in performing evolving searches.

3.2 Evolving Search issues

In chapter 1, the objectives of the research were discussed in terms of the different Evolving Search issues that would be addressed in this work. Of the evolving search interface issues mentioned in Bates's work [6], this work addresses the Evolving Search issues related to *opportunistic searching* (aiding users in browsing information), *information recall* (aiding users in retrieving previously viewed information) and *information visualization* (visualizing large amounts of data). The format of each subsection is as follows: each section introduces a particular interface issue and then presents, (1) the techniques available to tackle the issue, (2) examples of applications that use these techniques and (3) a discussion on the advantages, disadvantages and effectiveness of the techniques.

3.2.1 Opportunistic searching

Bates [6] suggested that an effective online interface allows users to browse quickly and randomly over search result information, much like the physical act of scanning over the titles of books on a shelf, or rifling through the pages of a book. However, the differences between the information structures of a library shelf and a list of search results means that random browsing cannot take place to the same degree. As stated in the previous section (section 3.1.1), search result information is normally presented in lists, each containing 10 results per page (although this default setup can be modified with various advanced commands). This means users cannot 'jump' between various parts of the list easily, and even scrolling a list of 100 results can be difficult. Designing an interface to accommodate this behaviour will benefit users in discovering new and interesting information, more so in Evolving Search tasks, where they need to gather as much information as possible, in order to further develop their information need and create a efficient query.

There are two methods to aid users in searching opportunistically using an online interface: (1) provide users with more information, so that they may 'jump' between the different parts of the data (like its physical equivalent, browsing a bookshelf), or (2) provide users with a method of sifting through large quantities of information quickly (like its physical equivalent, rifling

through the pages of a book). Looking at the first method, the most efficient way to provide users with more information is to abstract and simplify the data. Most search result and web page data comes in the form of text, which in turn occupies a large amount of space on the web page. Abstracting text often allows users to view more information on the screen, while still providing meaningful information. The following subsections will look at different abstraction techniques.

Technique : data reduction

Data reduction is an abstraction technique used to summarize textual information into its most salient keywords, reducing the screen space used, while still providing useful information, and filtering out unnecessary data. An example would be to reduce a document to its main section headings, allowing users to retain a sense of the document's structure, without the burden of having to read through the entire document. This also allows users to scan quickly over many such documents at the same time. Similarly this could be applied to a web page, summarizing the page as a list of it section headings, or as a specific set of meta-data (e.g. just showing the images on that page). All search engines abstract web page data into search results, reducing the web page to its title, a snippet of text from the page, and its URL. Some web browsers, such as Opera [98] allows users to reduce the page to just its hyperlinks through the use of modified style sheets, allowing the hyperlinks to be found more easily. Web browsers for mobile devices also perform data reduction (because of the small screen space) so that only the most pertinent information is displayed.

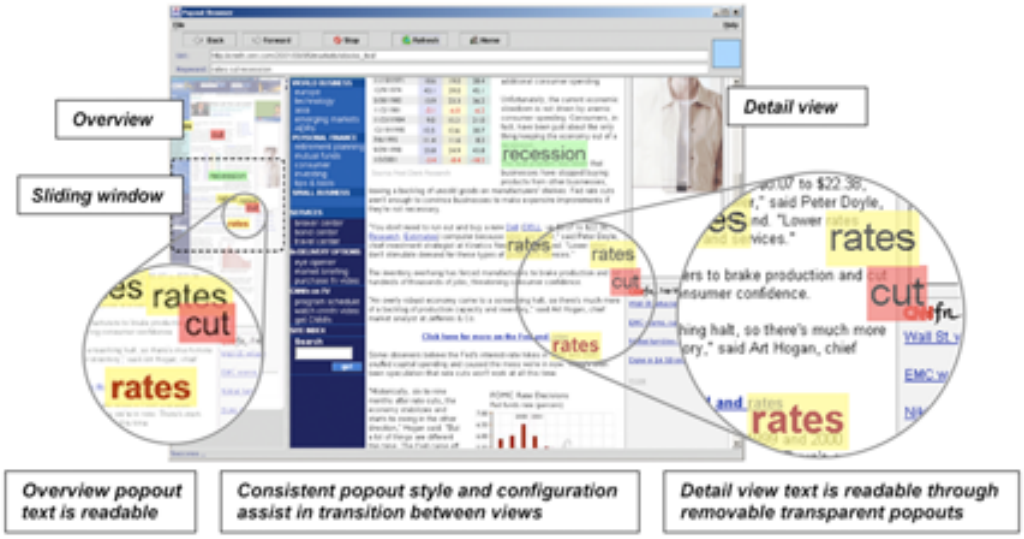

Figure 3.16: **Suh et al's [109] popout prism tool.** The popout prism tool was integrated as a sidebar (left) to a web browser tool. The tool displays the page at a lower magnification, allowing users to slide the tools 'window', to view different parts of the web page. Notice also the 'pop outs' (coloured boxes with text), which are words specified by users.

Technique : distortion and shrinking

Distortion and shrinking can also be used to abstract data, reducing the size of the information (text or images) in order to fit more information onto the screen. However, this can only be used up to a certain point before the information becomes illegible. This is more often used in images because, even with a lack of detail, users can often still match images, based on colour and shape. In Google's image search [36], thumbnails are utilized to display the images results returned. In the Thumbshots search engine [101], thumbnail pictures (of web pages) are provided for each search result returned. De-magnification can also be used to convey an overview of the data. An example of this is Suh et al's [109] popout prism, where a thumbnail view (of a web page) is placed side-by-side with its full magnification counter part (see Figure 3.16). This 'overview' of the web page allows users to navigate the page opportunistically, as its magnification makes the thumbnail large enough to be used as a map. While images are only slightly legible in the overview, the text is completely incomprehensible, and thus a special tool is used which can 'pop-up' text (see Figure 3.16) within the overview.

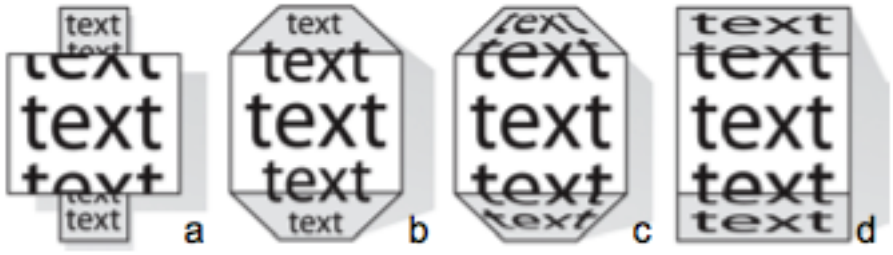

Figure 3.17: **Distortion geometry. (from Baudisch [7])** Different techniques which can be used to distort text to allow the viewing of both the data's focus and its context, each with its own advantages and disadvantages. The (a) Manhatten lens changes scale abruptly as the resolution boundary is crossed which can make some elements unreadable. (b) Zoomscapes can be used on single line objects or groups of objects but fails when used on very large objects. (c) The Central perspective shrinks content continuously with increasing distance from the focus. (d) Parallel projection only compresses off-focus content vertically and by the same factor

Technique : focus-and-context

Focus-and-context techniques, also known as detail-in-context and fish-eye views (after the fish-eye lens that distorts images) are an extension of distortion techniques. The technique, which was originally introduced by Furnas et al. [35] as part of his fish-eye views, increases the magnification around the data object that is currently being focused on, and shrinks the surrounding data, so that users are still aware of the context of the data. The amount of data in the focus can vary from a single object to a 'slice' of continuous data (a part of a web pages rendering). Similarly, the level of detail of the context can vary immensely (see Figure 3.17), sometimes the data is reduced to its smallest (unreadable) form, at other times there is a gradual change in size in the

context. The data presented can be of many different forms, such as plain text, images, web pages and data tables.

Applications using focus-and-context techniques utilize one of two major representation types, *continuous visual structures* ('unbroken' data such as maps, images, web pages) and *discrete visual structures* (data comprised of multiple ordinal parts, such as trees, graphs and search results lists). Applying focus-and-context techniques to a continuous visual structure is often described with a 'rubber-sheet' metaphor, where parts of the continuous image are stretched and distorted. Several examples exist, such as Leung and Apperley's [67] bi-focal display (Figure 3.18 - left of the figure) and Mackinlay et al's [69] perspective wall (Figure 3.18 - right of the figure). Applying focus-and-context to a discrete visual structure involves modifying the size and spatial location of various objects in the graph while keeping the relations between the nodes and links comprehensible. This gives an immediately recognizable and intuitive structure, examples include, Rao's Table lens [91] which visualizes tabular data, and Holmquist's [49] Zoom browser which utilizes his Flip-zooming technique (see Figure 3.19) to preserve the linear order, and emphasizes space-preservation over place-preservation.

Baudisch et al's Fishnet tool (a fisheye web browser) [7] built upon the work of Suh et al's [109] popout prism tool (previously mentioned). Instead of using Suh's side-by-side thumbnail overview, Baudisch et al. applied a focus-and-context technique to the main web page view, distorting the top and bottom of the web page to create an overview within the web page. As the scroll bar is moved, the focus is shifted, and the top and bottom parts of the page change to generate a new overview.

Technique : graphical display

Data can also be abstracted into a graphical display, which involves translating the data (e.g. textual data) into a graphical format, replacing the data-points with a pictorial representation that conveys the most important aspects of the data (e.g. representing pages or documents as icons). This technique is also used as a part of interfaces that detail the relationships between different pieces of data. For example, web pages represented as icons can be plotted onto a scatter-plot or connected into a node-link diagram representing a web page history. Graphical displays are not limited to two-dimensional views and can also be found mapped to three-dimensional views as well.

Applications using graphical display techniques often abstract data in order to increase the amount results displayed. The graphical form also allows relationships to be drawn between the different data points. The Grokker search engine [39] abstracts its search results into a hierarchy of nested circles (see Figure 3.20), allowing a larger number of search results to be displayed to users. Hasan et al's Hy+ browser [41] abstracts a browser's web page history into

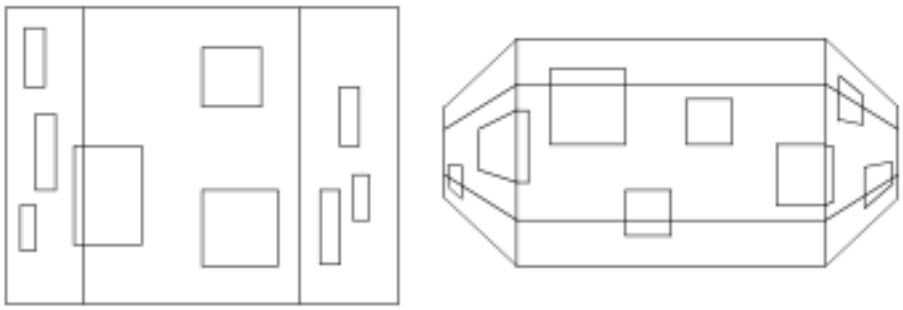

Figure 3.18: **Focus-and-context design (from [49]).** The bi-focal display (left) and the perspective wall (right). The Bi-focal display (based on work by [67]) compresses the horizontal portion of the material on the left and right (can also be done top to bottom - as in FishEye view [34]). The Perspective wall [69] uses graphical distortion with a three-dimensional metaphor showing the material as if it was receding into the background of the image on both sides of the focus.

Figure 3.19: **The Zoom Browser (by Holmquist et al [49]).** This tool utilizes the flip-zooming technique, to view pages of a document. Holmquist attempts to preserve the linear-ordering, while at the same time keeping the proportions of the pages, and using up the free space efficiently.

a two-dimensional node-link diagram (see Figure 3.21 and 3.22), with web pages represented by icons on the diagram. In Hy+, the relationship between the web pages in the browser history is made explicit, by linking together the web page (icons) with lines, mapping the path of users progress.

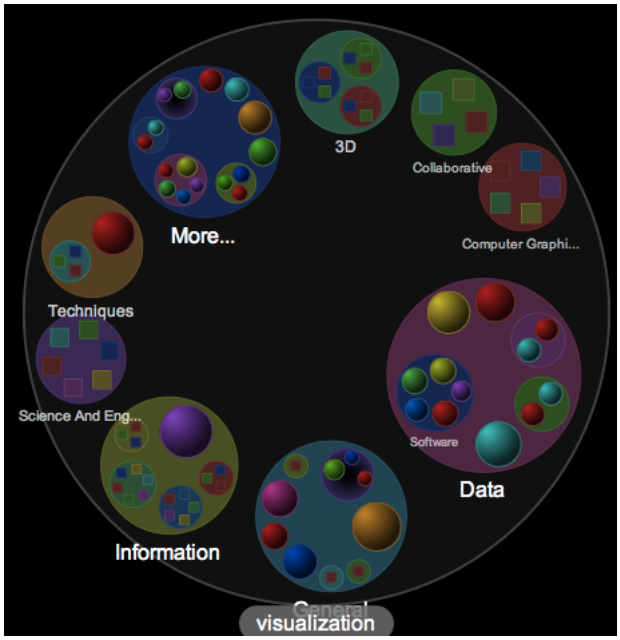

Figure 3.20: **The Grokker web browser (from the grokker website [39]).** When a search is made (e.g. web visualization), the results are split into several categories (e.g. information, data) and sub categories (e.g. software). These are represented by a set of hierarchical circles. Investigating the categories (by clicking on the circles) zooms users into the circle, revealing more circles (sub categories) or displaying actual web page icons. The size of the circle is the visual cue, denoting the amount of information (web pages) contained within that category.

Three-dimensional display techniques allow more data to be displayed, through the use of an extra dimension as well as allowing the use of novel interactions techniques (e.g. three-dimensional worlds). NIRVE's [24] three-dimensional scatter-plot takes advantage of the third dimension, by allowing a third variable to be compared in the display. Certain three-dimensional interfaces allow the introduction of completely new interaction techniques, such as Card's [19] WebBook, which generates a three-dimensional book which users can 'flip' through (see Figure 3.23). The WebBook preloads a collection of web pages into a three-dimensional 'book'. There are several ways to 'flip' through the book depending upon the user's actions. Clicking on a hyperlink leading to an internal page will flip the view to that section, clicking on a page itself (not a link) will flip to the next page, and clicking to the far left and far right will advance or decrement the page (the relative distance of the click indicates how far to jump). Users can also scan using the backwards and forwards buttons.

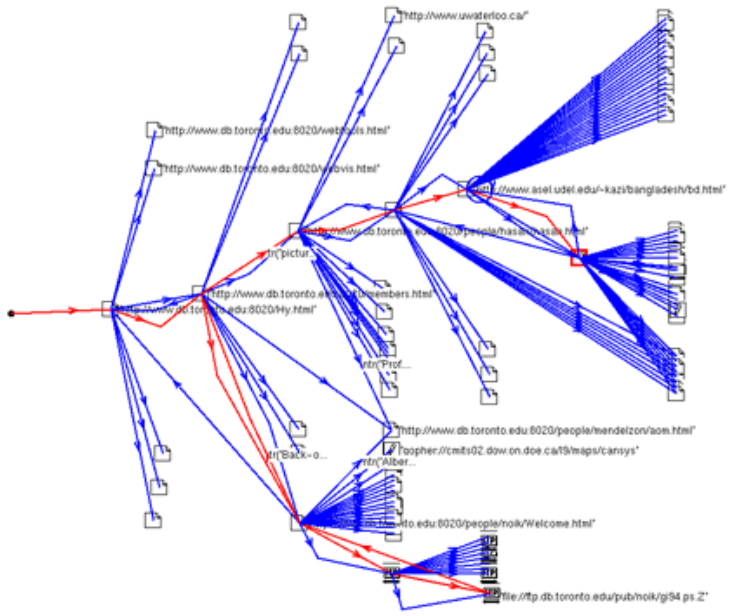

Figure 3.21: **Hy+ Web Browser.** Hasan et al. [41] designed a two-dimensional, graph based visualization that displays users progress through a website. As users view web pages, nodes representing web pages and their connecting edges are added to the graph, as well nodes representing hyperlink destinations from each of the web pages. The Blue edges represent potential travel paths, the edges in red show paths users have visited. Clicking on any node will take users directly to that web page.

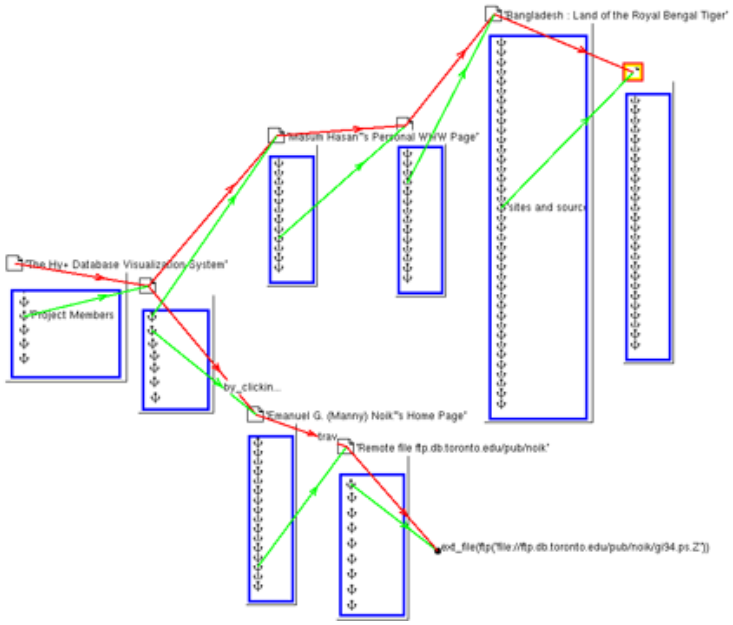

Figure 3.22: **Hy+ Web Browser ('Blobs' view).** A different view of the above Hy+ browser, also by Hasan et al [41]. Users can create a 'box' (called a 'Blob') to group together sections of web pages that are located close to each other. This allows users to prune large parts of the tree allowing a less crowded overview of search progress.

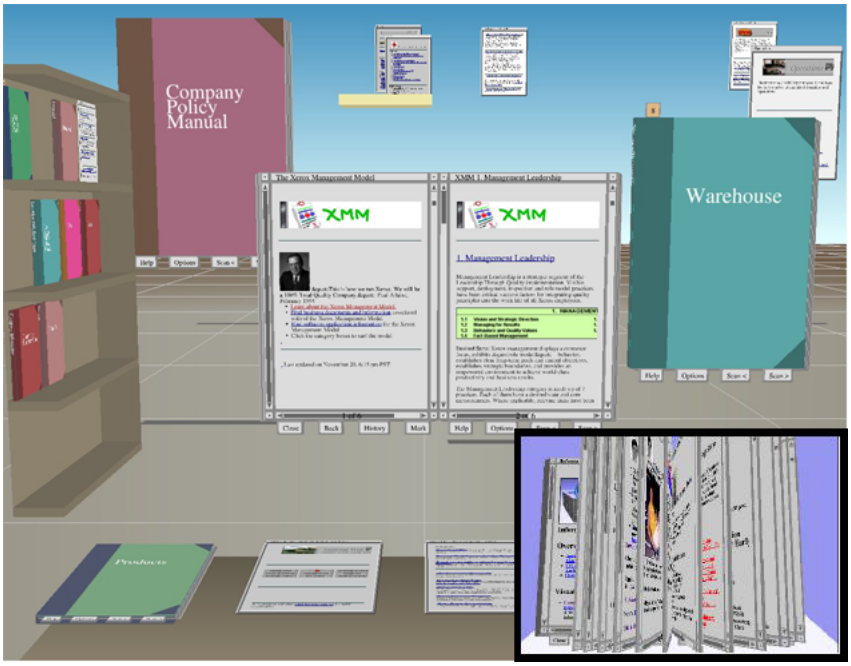

Figure 3.23: **Web Forager and Web Book.** The WebForager workspace (developed by Card et al. [19]) creates a three-dimensional desktop from which users can search for information. The data is represented by three-dimensional books, that can be opened and read. These books are in turn stored on a three-dimensional representation of a bookshelf. The WebBook, is a tool for storing web pages, and like a real-life book, its pages can be turned. The pages of the Webbook can be seen 'flipping' in the inset on the right.

Discussion

The techniques that have been presented demonstrate how current researchers have provided solutions to opportunistic searching, through the use of data abstraction. Although many of these techniques have been successful, they only represent part of the solution, to the many challenges of evolving searches. The advantages and disadvantages of the techniques will now be discussed. While data abstraction allows users to view more data on the screen, contextual information which may have been useful to users is lost. This is also apparent in both data reduction and distortion, for example, by reducing the amount of data users might miss out on valuable keywords, and by distorting a piece of text or an image it becomes difficult to read or view it. This loss of context also takes place in tools which use magnification, for example the NaviQue [35] search result visualization which uses a zoomable interface. When users zoom-in to inspect results, they completely lose the surrounding results which form the context of the search. The focus-and-context technique reduces this problem by allowing users to focus on a result while still being aware of the data's context.

Looking at Baudisch et al's Fishnet tool [7], this particular method of focus-and-context (which distorts the web page rendering), gives no added benefit to users since the information in the context becomes too small to read, and any images in the context will be distorted. However

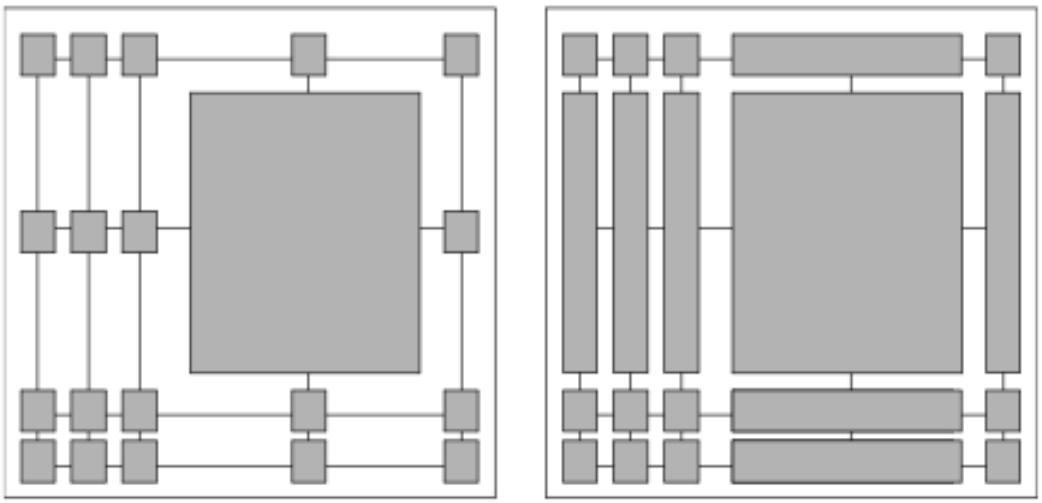

Figure 3.24: **Wasted space v.s. distorted space (from Holmquist et al. [49])** Different styles of focus-and-context visualizations, create different kinds of problems. Left, the proportions are preserved, but there is wasted space, right, the space is used effectively, but the objects are distorted.

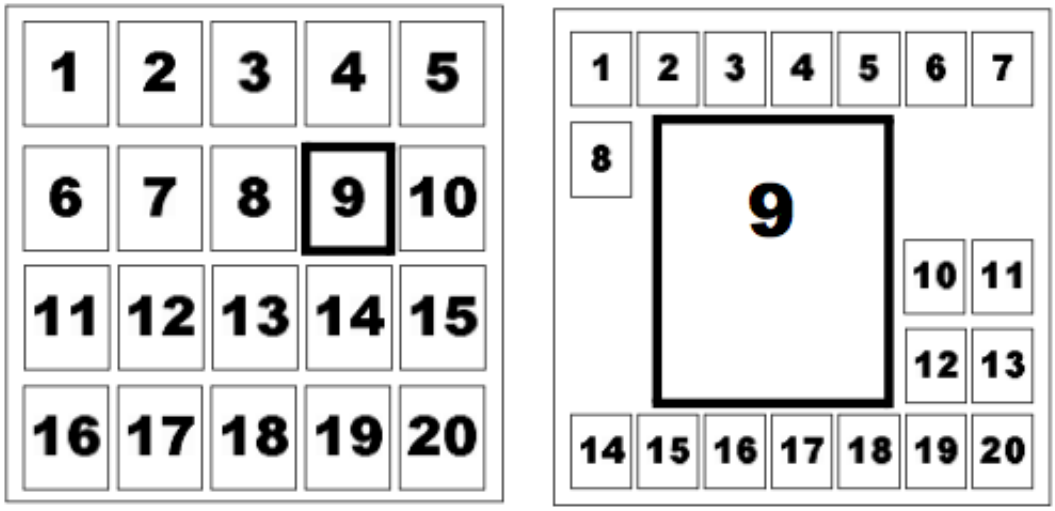

Figure 3.25: **Flip zooming layout.** This technique was developed by Holmquist et al. [49]. Flip-zooming starts with several objects laid out, equally spaced on a two-dimensional surface (using thumbnails), when a result is selected it increases in size, while its surrounding elements are miniaturized and reorganized around it. Notice that their linear-ordering is also preserved (to a certain degree).

the tool also has a 'find function' that highlights search terms specified by users, made visible in the context of the view through highlighting and enlargement of text. This allows them to locate the terms they are interested in, outside of the focus. Compared to the previously mentioned popout prism tool, Fishnet has different strengths and weaknesses. Unlike the popout prism tool, Fishnet makes full use of the available space, however, the Fishnet tool has the problem of an inconsistent aspect ratio in its context views, as well as a greater compression ratio (and hence less detail) than in the popout prism tool. Additionally, this modification of the visual structure of the data can make it distorted and unreadable. An experiment conducted by Baudisch et al. [7] showed that the popout prism tool and Fishnet view performed well in different circumstances, and they came to the conclusion that it would be ideal to provide users with both options when searching web pages.

Applying focus-and-context techniques to an interface has its own set of disadvantages, as can be seen in Figure 3.24. In attempting to preserve the proportions of the data objects, a large amount of wasted space is generated (see Figure 3.24 - left). If the tool is designed to make the best use of space possible, then the data will lose its proportions and become distorted (see Figure 3.24 - right). Holmquist et al. [49] attempted to solve these problems with minimal wasted space, and minimum distortion. They realized that by attempting to preserve the 1-dimensional linear-ordering of the data, they were preserving the 2-dimensional ordering as well, and as a result decided to break the rigid frame imposed by the ordering while still preserving the linear order.

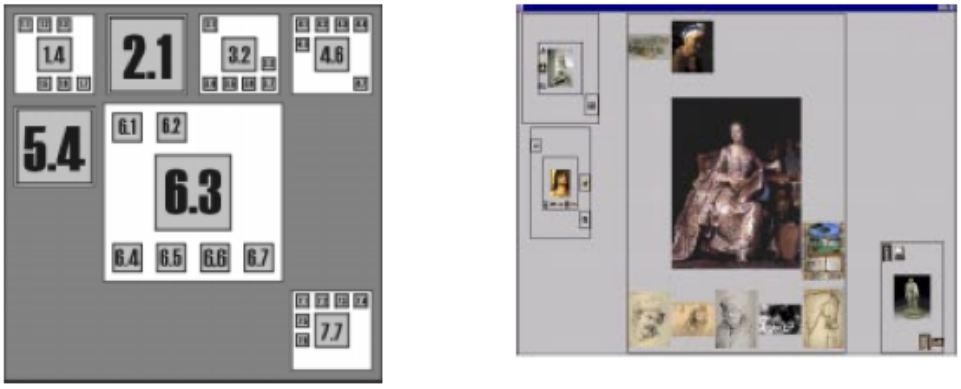

Figure 3.26: **Hierarchical image browser (Holmquist et al. [49]).** Left, is an outline of a modified flip zooming layout technique which divides the data into specific collections. Right is an example view of an image collection utilizing the layout technique. This is a place-preserving design, the focused object is always central to the view (never moving), with objects forming clusters around the focus.

One method attempted by other interface designers was to place results (search results in this case) in a *spiral*, as seen in Spoerri's Rank spiral [105], and Cugini et al's [24] three-dimensional

spiral. Although both interfaces preserve the ordering of the search results, the closeness of the results infer a relationship (that they are similar) that does not exist. No attempt has been made to apply focus-and-context techniques to such a spiral method. Holmquist [49] suggested a technique called *flip-zooming* (see Figure 3.25), which preserves the linear order. However, the design emphasized space-preservation over place-preservation (location), which can be troublesome to users, whose view will change drastically every time a result is clicked on.

Holmquist describes three implementations which use flip-zooming, the Zoom Browser (for text documents, see Figure 3.19), the Image Browser (for image collections), and the Hierarchical Image Browser (see Figure 3.26). Ideally there should be no distortion of the data and no wasted space, however this is only really achievable with certain visualizations such as the Table lens [91], which deals with tabular data. Table lens uses a standard table format, but applies focus-and-context on selected columns and rows. Unlike the bi-focal display and perspective wall, none of the data is distorted, and no empty space is generated, because all data is only viewed in one of two states, numerical or graphical.

Using a graphical display reduces the data's size (through abstraction) while allowing users to avoid the problems of distorting the data. However, abstracting the data this way often removes specific details, as well as the context of the information which can be troublesome for users. Abstracting the data can also make it difficult to interpret, for example if the graphical display uses a representation that users are not familiar with. Attempts have been made to create interfaces based on more familiar metaphors, for example, the aforementioned WebBook [19] mimics a physical real-life book. The problem with WebBook is that it seems to discard the website's entire structure, organizing the website into 'chapters' and 'sections'. Many websites cannot be ordered into such a neat structure, and certainly not into a linear-ordering. It is interesting to note that the use of analogies (such as the WebBook) was supported by Bates [6] as a means of helping users better understand interfaces and allowing them to perform search actions which more closely resemble their real-life counterparts. The next section will look at information recall, the second evolving search issue addressed in this work.

3.2.2 Information Recall

Bates [6] suggested that the ability to quickly recall previously visited data sources, would be useful during a search, since it can be easy to lose track of specific information among the large quantities of data generated. In an evolving search of online sources this is especially true, given the amount of multiple searches generated, and web pages viewed.

Studies of web searching patterns by Tauscher et al. [114] detailed two important facts about information recall, that most web pages viewed by users are revisits, and that web browser history tools are rarely used. Tauscher et al's studies showed that 58% of the web pages that

users visit during a browser session are revisits to previously-viewed pages. This may be because users return to pages that are frequently updated (e.g. News web pages) or have a specific function (e.g. search engine and dictionary web pages). They also noted that while many pages are revisited, users continually incorporate new pages into their repertoire at a regular rate.

Tauscher et al's research also showed that on average, the majority of web pages that users visit will only be viewed once (60% of web page visits) or twice (19 % of web page visits). During a search session, the back-button usage comprised of 30% of all navigation actions, the bookmark counted for 3% of the actions, where as the history tool provided 1% of all actions. The results of Tauscher et al's [114] studies indicate that users often recall web pages, but very rarely use either the bookmark or history function to retrieve these pages. This implies that users either recall web pages from memory, or simply re-perform searches on a regular basis to retrieve information. There are two principle methods of storing data on a web browser to be recalled later, these are *automatic history systems* (such as the tools browser history), where all the web pages that users view are recorded, without needing their input, and *manual history systems* (such as bookmarks), where users explicitly stores selected web pages.

The browser history is an automatic history system commonly found in web browsers. The tool records all web pages viewed by users over the course of their search sessions, represented as a temporally ordered list. Other automatic history systems exist, and are integrated into other components of the web browser, for example users can access their recent web page history through drop-down boxes in both the address bar and the back-button. Automatic history systems don't need specific input from users since they record every page that is visited, thus if users forget to mark specific web pages, they can always use the search history to go find it. However finding the information within the search history will take longer than if it was bookmarked, because the web page list needs to be searched.

The bookmark tool (sometimes called the 'favourites') is a manual history system often found in web browsers, and allows users to store specific web pages in an organized fashion. Bookmark tools have become more advanced over the last few years and now keywords and notes can be associated with specific bookmarks, allowing them to be searched. Some web browsers have bookmark toolbars (such as Mozilla's Firefox [32] and Apple's [89] Safari web browser), which allow quick access to frequently used bookmarks. Problems exist with current manual history tools. With bookmark tools, if users forget to bookmark a page after visiting it, then once they pass the page, they cannot easily recall it. Users must also take the time and effort to organize their bookmarks and keep them in order, and up to date.

Current history tools are unsuitable for evolving searches, they are cumbersome to use, hard to search, and hard to organize, all of which add to the difficulty of recalling relevant information in a timely manner. The rest of this section outlines the problems with existing search history

tools with regards to the evolving search, the different techniques that can be used to represent a search history, examples of applications using search history tools, and conclude with a discussion on the issues related to designing history tools, as well as a set of criteria for efficient history tools.

Due to the failings of the designs of current web browser history tools, alternate methods of recalling web pages have been implemented by researchers. In the case of *automatic history systems*, emphasis has been placed on finding ways to aid users in searching for specific web pages among the large list of automatically recorded URL's. In the case of *manual history systems*, emphasis has been placed on creating tools that allow users to quickly and easily mark and retrieve specific data objects. Each of these will be discussed in turn.

Technique : automatic history tools

The simplest method of searching a large list of URLs would involve some method of filtering that will screen out unwanted data based on a set of criteria. The most common method is to use a search box, in which users input some keywords. These keywords are then checked against each bookmark's meta-data tags (e.g. URL, Title) and the list is modified to contain only those results that match the criteria. This is not always ideal, given the fact that users remember web pages in different ways (e.g. colour, images, keywords in the page), and thus they may not remember the web page by the information used to store it. Showing the trail of web pages users visited during a search can help them remember the position of the information they are looking for. However following a trail using the hyperlink highlighting is slow, since users have to move slowly, step by step through the visited web pages. Ideally, a graphical overview should be used, allowing users to jump from one web page on the trail to another instantaneously. Certain graphical overviews (such as two-dimensional tree structures) automatically bypass the branching problem (discussed earlier in the chapter).

There are tools which utilize two-dimensional graphical overviews to display automatic history data. The WebNet [22] history view uses a graph-like layout (see Figure 3.27 - left), with web pages represented as interconnected circles. Mosaic G [4], uses an hierarchical layout (see Figure 3.27), representing web pages as blocks on two-dimensional space. PadPrints [47], based on the Pad++ zoomable interface, lays out the history data as a hierarchical tree. Like Mosaic G, PadPrints also displays web pages as blocks as well as providing titles and thumbnail images of the web pages, but differs in that it provides a fully zoomable interface with scalable objects (branches that are no longer of interest can be shrunk until they are a less significant size). Trail highlighting can also be used as part of a two-dimensional graphical interface to illustrate the path taken by users. For example Hasan et al's [41] Hy+ browser, (see Figure 3.21) uses a two-dimensional graph to map users progress when browsing web pages, as well as showing

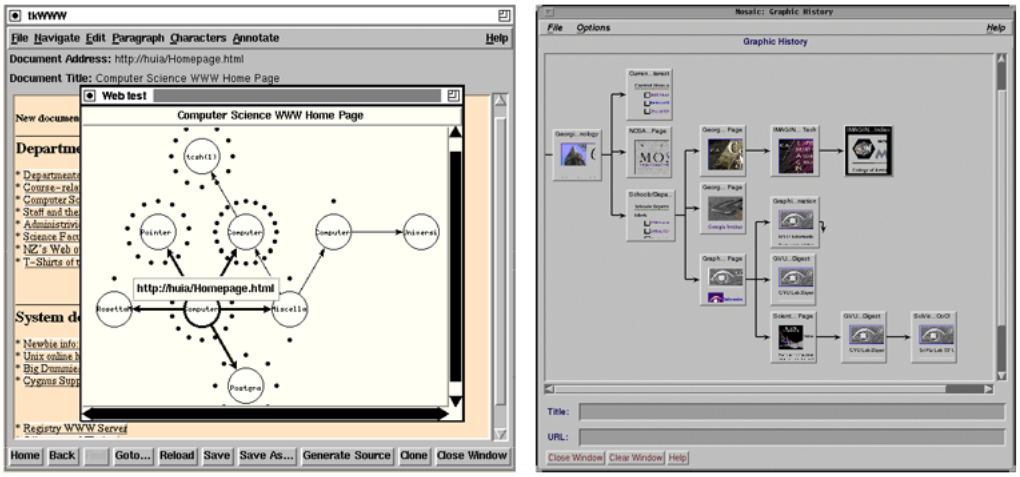

Figure 3.27: **Graphical history navigation.** Left - WebNet [22], Right - Mosaic G [4]. WebNet displays the browser history in a separate window containing interconnected circles representing web pages, surrounded by small black dots representing links out of that web page. WebNet contains a filter that can alter the size of the nodes based on a set of different criteria including frequency of visits, recency of visitation, and distance from the current displayed page. Mosaic G represents web pages visited as squares, detailing the title, url and thumbnail image of the web page. The layout is that of a two-dimensional hierarchical tree (left to right). Users can reduce the view by zooming out of the map, and can also condense branches of trees that are no longer of interest.

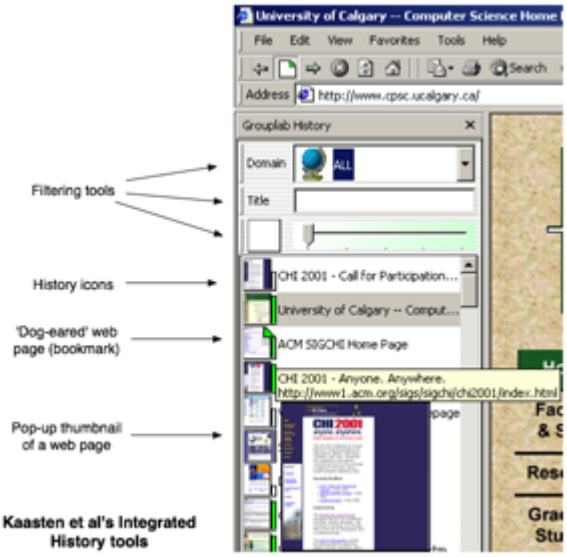

Figure 3.28: **Kaasten et al's Integrated History Tools.** Appears as sidebar on Internet Explorer. Each web page is represented by an thumbnail icon of the web page, with number of views represented by a green bar on the side of the icon. Pages which are explicitly bookmarked are 'dog-eared' (have a small triangle in the corner, like folding a page corner). The icon ordering is based on a combination of recency, number of visitations, and whether it has been explicitly bookmarked. Filtering tools are available to filter the history based on recency, title or domain.

paths to adjoining unvisited web pages. In Hy+ the paths that users have visited are highlighted in red, while those which are unvisited are coloured in blue.

Technique : manual history tools

Techniques for retrieving manual history data involve either placing pointers to the data in plain sight where users can quickly access them e.g. on a bookmark toolbar, or by highlighting the data so that it can be found among large amounts of data easily. The suitability of the technique varies depending on the interface. For example in some graphical interfaces, it is more suitable to highlight a data point than store it in a structure to be retrieved, because the interface already allows users to jump to any point in the data with a single click. The Insyder tool by Mann [72] contains a two-dimensional scatterplot view in which the plotted symbols can be highlighted in a specific colour to make them stand out. This permits users to quickly and easily recall the requested page and compare the data with the tool's different views. Card's 3Book [18] uses a three-dimensional book for its interface. The bookmark takes the form of a coloured strip (much like a physical bookmark) which users can see clearly, and can be selected to open an associated page.

Technique : manual and automatic

Some tools exist that combine both automatic and manual history techniques. Kaasten et al. [61] implemented an integrated tool for searching web histories and bookmarking (see Figure 3.28), which worked as a sidebar in the Microsoft Internet Explorer. The sidebar is dominated by a list of icons each of which displays a miniature thumbnail pictures of the website it represents. At the side of each icon is a small green bar, with a height determined by the number of visitations that have been made to the page. Icons are added to the tools list as users search, and web pages that are visited frequently automatically get moved to the top of the list. Icons which users think are relevant can be bookmarked, which will keep them at the top of the list. Users can filter the history list based on recency, domain or title keyword using a combination of slider bar and search box controls.

The tool's thumbnail icon shows a unique representation of the web page, which is not legible, nor useful for identifying specific meta-data about the page. Kaasten et al. [61] solves this problem by allowing a hovering pop-up of the page at a larger resolution, as well as its title and URL, when the cursor is hovered over the thumbnail. Users can also customize the icon's label, so as to give it a more meaningful title. One of the major differences between this tool and other search history tools is that the history stores data on a recency based system, as opposed to the traditional stack based system, thus avoiding the branching problem (see [61]).

The use of recency based systems is echoed in other research. Berkun [11] described a

bookmarking system that would store information on how frequently websites were visited, and display the most frequent web pages in a toolbar or other easily accessible object in the browser. He called this 'Intelligent Bookmarking'. Berkun also specified that such bookmarks should be automatically checked for broken links and duplicates in order to keep them from becoming unmanageable. The Opera web browser implemented a form of intelligent bookmarking, where bookmark drop-down lists were trimmed, showing only new and frequently accessed bookmarks, and hiding bookmarks which were used infrequantly.

Discussion

Through a series of experiments Tauscher et al. [114] observed and developed a set of criteria for the design of history systems. However many of these criteria are out-dated, since they have become commonplace in browsers. For example, Tauscher et al. noted that records of all URL's should be kept, since some history lists in the past were only kept on a session by session basis. Nowadays users can specify exactly how long histories should be kept, and how much space they should take up. Tauscher et al. also noted that grouping URLs into web tasks would be useful for pages that people revisit regularly. Modern web browsers can often save entire sessions of web pages (including their histories). The previously mentioned Webbook (Card et al. [19]) also grouped together web pages, into three-dimensional representations of books, which users could view. Below is a summary of the design theories that are still valid:

1. **Recalling URLs should be easy.** Recalling web pages via a history mechanism should be easier than simply calling up a search engine and performing a search. There are still many pages that users revisit that they are not aware of (see Tauscher et al's [114] experiment), and an automatic system to suggest pages to revisit based on revisitation and recency would be advantageous.

2. **History lists are short.** Tauscher et al. stated that a lengthy history list is unlikely to be worthwhile, considering the high cost of screen real estate and the cognitive overhead from scanning so many results. They estimated that a size of 6-10 results would be adequate based on the fact that the experiment [114] showed that 43% of all revisits were from the last 10 results, with this percentage only growing marginally when the result's range was doubled and tripled.

3. **History should include both recency and revisitation.** Tauscher et al. found that while recency was a major factor which affected revisitation, many results were revisited that were not used in the recent past. They suggested combining recency with amount of revisitation.

4. **Representation should be meaningful.** It was discovered that titles and URLs are not always good indicators for the contents of a web page. The use of thumbnail images was suggested to represent pages, since users who have already visited the page will recognize the layout, colors and structure of the web page. However recognition is determined by thumbnail size, and larger (more recognizable) thumbnails take up valuable screen space.

5. **Customization.** Users should be given the opportunity to make significant results even more distinct, using their own names, icons and groupings, all of which will aid quicker recall.

As mentioned previously, efficient recall systems are needed when performing evolving searches. This section has looked at the inadequacies of current information recall systems, and discussed the different techniques and tools that could be used in their stead. The next section will look at the third evolving search issue of Bates [6] that this work addresses, information visualization.

3.2.3 Information visualization

Bates feared that evolving searches would generate large amounts of information, and it would be difficult to view all the data at the same time, on the same screen. The solution she envisaged was to increase the resolution and size of the VDU. Since that time, technology has indeed improved greatly, but still problems remain. This section proposes a software based solution using visualization techniques that transforms large amounts of data into useful and meaningful displays, beginning with a look at the different techniques that are available to display the data, and then moving onto a few examples of different applications utilizing these techniques. The section concludes with a discussion of the different advantages and disadvantages of each of the techniques as well as their suitability for visualizing evolving searches.

Note that there is some overlap in terms of techniques described between this section, and the aforementioned section on *Opportunistic searching*. While both sections involve tools for visualizing large amounts of data, those in the Opportunistic searching chapter concentrate on the different interaction techniques involved with manipulating the data, while this section deals primarily with the techniques involved in displaying the data. As mentioned in section 3.2.1, one method of providing users with more information in a restricted screen space is to abstract the data. Previous sections have discussed the different abstraction techniques available, graphical displays were identified as an effective method of abstracting large amounts of data. In the research community it is well understood that information displayed in a graphical form can be more quickly assimilated than its textual counterpart (see Shneiderman [100]), and trends and relationships between different parts of the data can be more easily discovered.

Graphical displays of search result data come in a variety of different forms. Some use

simple and easily understood representations such as bar charts or scatter plots, others use more complicated representations, such as the tilebar [45], spiral [105], and topographical map [14] visualization techniques.

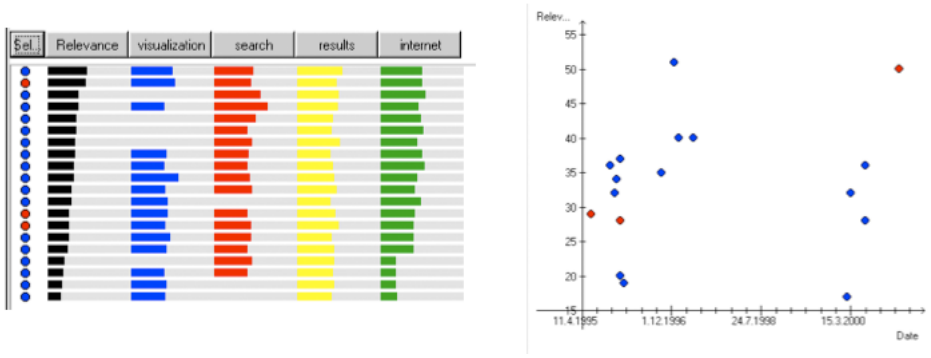

Figure 3.29: **Mann's [70] Insyder Tool.** Two of the views of the Insyder tool, the Bar chart view (left) and scatter plot view (right). In the bar chart, the results are ordered by overall relevance, shown in the first column. The remaining 4 columns represent the different terms used in the query, and their corresponding relevances.

Technique : bar chart

Mann [72] employs a bar chart view as part of his Insyder tools (see Figure 3.29). Multiple bar charts (displayed as side-by-side columns) are used in the display, each bar chart representing a list of search results, and each bar representing a search result. The length of each bar in the list is dependent on the variable associated with the column e.g. the far left column (relevance) maps the bar's length to its search result's relevance ranking, while the search result bars on the far right (internet) have lengths based on the number of occurrences of the word 'internet' in each search result.

Technique : scatter-plot

When search results are plotted onto a scatter-plot, meta-data is mapped to the x and y axes, and extra meta-data is often encoded into the data points on the scatter plot, using various perceptual variables such as color and shape. The Multiform Glyphs tool by Boukhelifa et al. [93](see Figure 3.30) uses a scatter-plot as part of its coordinated multiple views visualization. The tool has four coordinated windows, but the view of interest is the domain glyph view which maps search results onto the scatter-plot using the internal and external link count for the x and y axes respectively. Note that the shape of each glyph varies depending on the extension of the web page it represents (e.g. circle for .com, square for .edu). Cugini's [24] three-dimensional scatter-plot (see Figure 3.31) extends the original two-dimensional scatter-plot, mapping user-specified keywords to each of the three axes of the scatter-plot, allowing users to discern relations

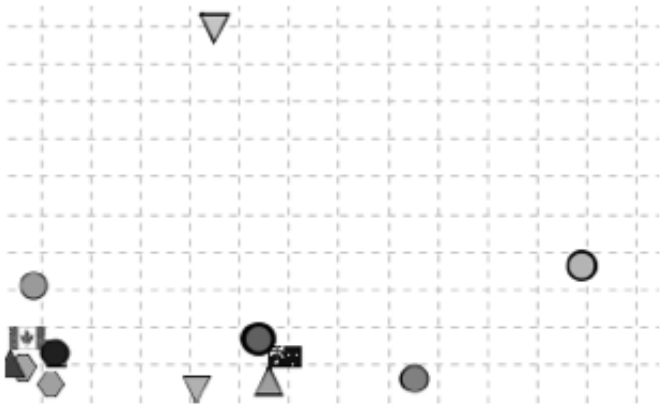

Figure 3.30: **Multi-form Glyphs. (by Boukhelifa et al. [93])** Each glyph in this view is represented by a shape based on its domain suffix (e.g. circular for .com, square for .edu, octagon for .net), the ranking of the web page is mapped to the color of the icon, and the width of the icon outline represents the pages size. The X and Y locations of the icons are based on the number of internal and external links of each result.

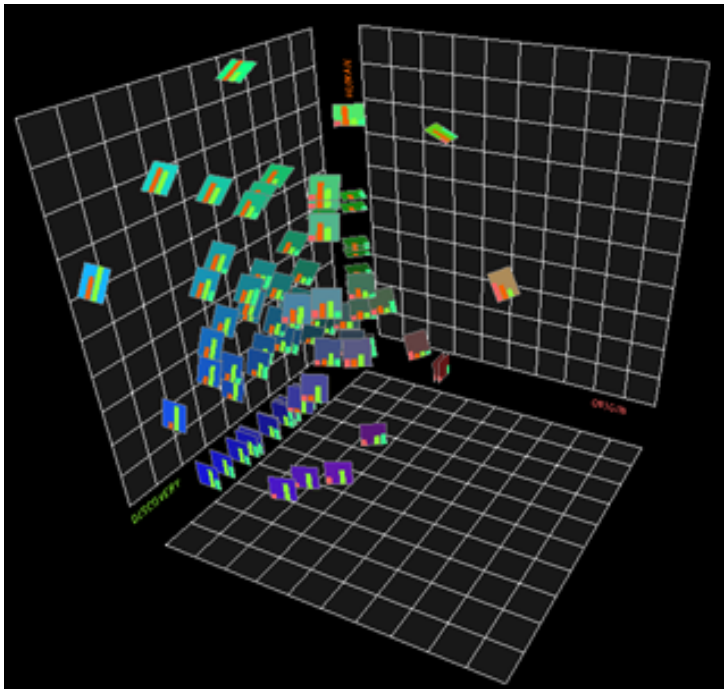

Figure 3.31: **Cugini's three-dimensional scatter-plot [24]** places search results in a three-dimensional space, the axes representing the result's relevance to specific terms. Each search result is represented by a small bar chart icon, which is a summary of the information on the three terms used to calculate the result's coordinates.

between the search results based on their keywords. The data-points are represented by icons, whose perceptual variables are altered to visualize more meta-data; in this case a miniature two-dimensional bar chart replaces the standard data-points.

Technique : Tilebar

The Tilebar is an interesting technique developed by Hearst et al. [45] (see Figure 3.32), which bears resemblance to the bar chart technique. The tilebar is a method of visualizing the concentration of particular words within a web page. In its most basic form, a tilebar is a rectangular block divided into several other smaller rectangular blocks. The larger rectangular block represent the entire web page, and the smaller rectangular blocks (within the larger blocks) represent different parts of the web page. Each block is assigned a shading which is based on the concentration of a particular word in that section (darker blocks mean a higher concentration of occurrences of that word in that section).

Technique : spiral

The Spiral technique was introduced as a solution to the problem of fitting data with a linear order, into a fixed space (see Figure 3.33). By winding the line of data around into a spiral, the linear-ordering is preserved, and all available space is used. Like the aforementioned scatterplot examples, the data-point in the display supplies additional meta-data to users, through the use of different perceptual variables (e.g. using colour/size/distance to convey additional information). Spoerri's MetaCrystal [106] contains a rank spiral view (see Figure 3.33), where each data-point has extra meta-data encoded into two of its perceptual variables, shape, and colour. The different shapes represent the number of searches that the data-point exists in (e.g. pentagon = five searches, square = four searches) and the different colours within the data-point show which searches the data-point exists in (e.g. red = google.com, blue = teoma.com). Cugini [24] created a three-dimensional spiral as part of the NIRVE interface, which visualized search results, where the data-points were replaced with bar chart icons representing the number of user-specified words present in the search result.

Technique : topographical map

The Topographical map was originally used to map geographical information, but was extended towards visualizing other types of data. Topographical map techniques are a popular choice, because users are familiar with consulting maps for navigation. Most map interfaces utilize the peaks and troughs present in geographical maps to point users to interesting information and away from non-relevant information respectively. A number of two-dimensional topographical

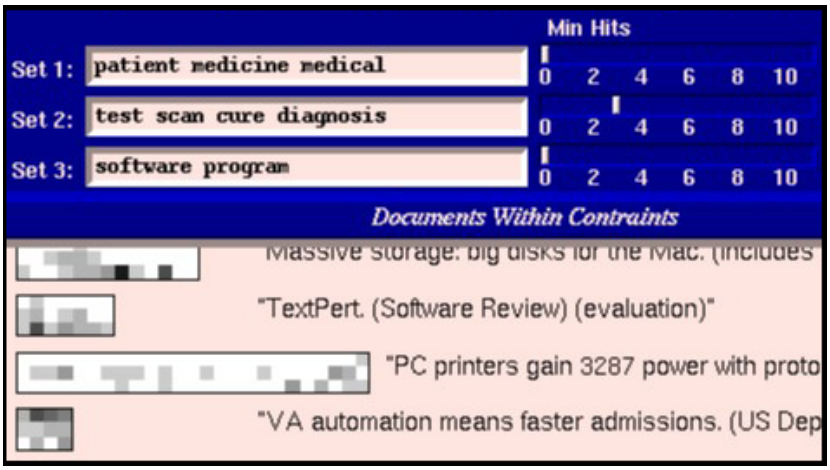

Figure 3.32: **Tilebar search interface (by Mann [72]).** A tilebar is a rectangular block divided into several other smaller rectangular blocks. The larger rectangular block represent the entire web page, and the smaller rectangular block represent different parts of the web page. Each block is assigned a shading which is based on the concentration of a particular word in that section (the darker means a higher concentration of occurrences of that word in that section).

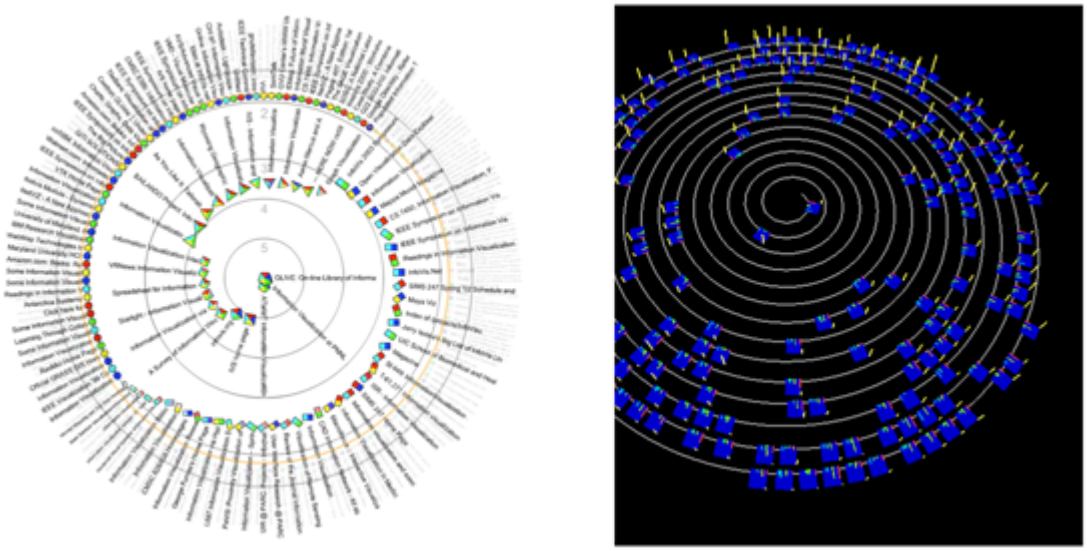

Figure 3.33: **Spiral metaphor visualizations.** Spoerri's [105] rank spiral (left) & Cugini's [24] three-dimensional spiral view. Often used on linear data to fill the area space. One problem with this format, is that users may unintentionally misinterpret the relations in the data. Since the icons in the different coils of the spiral are so close together, users might mistakenly assume a similarity between results in different coils, where none exists.

map visualizations exist. Cartia's newsmaps [80] (see Figure 3.34 - left diagram) is a self organizing map that clusters similar search results together onto a topographical map. 'Peaks' form, where large amounts of similar documents gather together. WebSom [51], by Honkela et al., utilizes a self-organizing map algorithm to cluster data on a two-dimensional map, (figure 3.34 - right diagram), and utilizes the color orange at different intensities, where darker patches of color are used to represent particularly large clusters of information. Themescape, developed by Miller et al. [77], uses document abstracts to generate three-dimensional topographical maps (see Figure 3.35 - left). Similar documents are clustered together, and these clusters are translated into three-dimensional peaks. It is a simple step to associate the height of the generated 'mountains' with clusters of useful information. Boyack et al. [14], in their VxInsight tool (see Figure 3.35 - right), used much the same geographical representation to view similarities in bibliographical data.

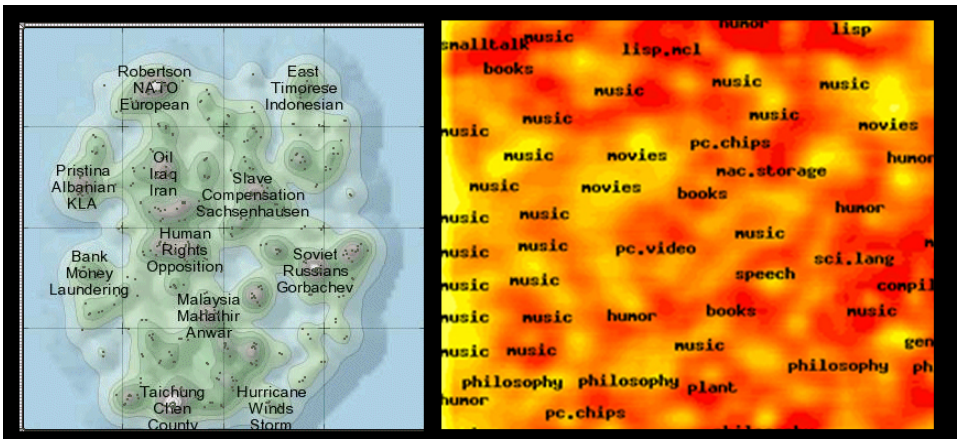

Figure 3.34: **Topographical maps visualizations.** On the left is the Cartia Newsmaps tool [80], which visualizes search results onto the map, grouping the results according to similarity to form 'mountains'. Right is Honkela et al's WebSom [51] which also maps similar search results together, but instead generates concentrated patches of colour where similar results gather.

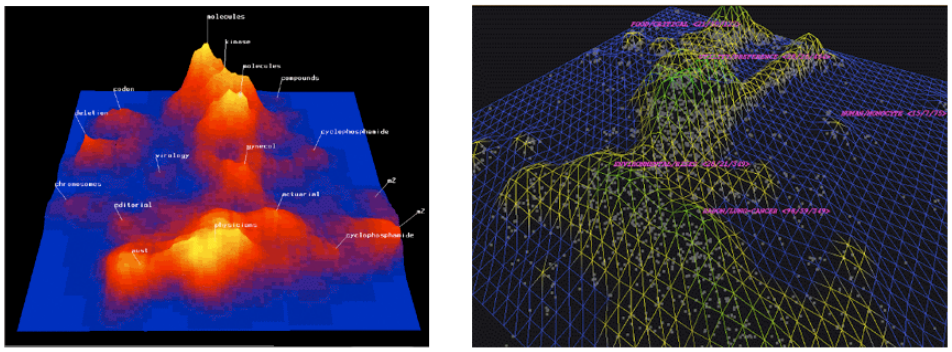

Figure 3.35: **Topographical map metaphor.** On the left is Themescape (by Miller et al. [77]) and on the right is VxInsight (by Boyack et al. [14]). Color and height are used in combination to show clusters of different subjects.

Discussion

The single problem that affects all of the mentioned techniques is the loss of textual information. Many of the techniques attempt to map various meta-data into the data objects using perceptual variables. However, not all meta-data is useful (e.g. users might not be interested in the number of internal/external links in a web page) and the relevance of meta-data can vary from search to search. Another problem with graphical interfaces is the intuitiveness of display, for example a scatter-plot is a common occurrence in statistics, but less so in web searching, which can cause some confusion over its purpose and use. Most graphical displays (with the exception of the spiral technique) break up the linear-ordering of the search result list, in order to allow users more freedom in traversing their search results. This can be confusing for users who are used to the linear-ordering of the search results, and can sometimes make searching for information more difficult.

An integral part of any graphical display, is the mapping of data into graphical objects. Often the forms of the mapped objects directly symbolize the original information e.g. icons drawn as books infer to users that the information contained by the icon is textual in nature, and contains part or the whole of a book. Furthermore there are different perceptual variables that can be used in conjunction with graphical objects to convey quantitative and qualitative information to users, such as size, color, distance, intensity and shape. Each perceptual variable has different advantages and disadvantages e.g. distance can be used to indicate similarity between two objects, but can also implicate relations between objects which are located in close proximity by chance. Developers utilize these different perceptual variables to bring meaning to abstracted data e.g. associating object size with page count.

3.3 Online interface issues

When Bates [6] first wrote about the evolving search over ten years ago, the WWW was still in its infancy, and as a result most of the issues she discussed referred to database and library interactions. However, the WWW introduced new problems for searchers. This section looks at two general online interface issues which affect the evolving search, these are the issues of *Information-seeking* and *Information management*.

3.3.1 Information-seeking

Previous sections have looked at different display techniques for dealing with large amounts of data (information visualization), and different interaction techniques for moving through the data easily (opportunistic searching). However, the needs of users vary greatly, and having the capability to view the data and move through it easily does not guarantee that users will

be able to find interesting information relevant to their search. This is best summarized by Shneiderman's [100] famous quote *"Overview first, zoom and filter, then details on demand"*, where the 'overview' refers to the ability to visualize the information holistically, 'zoom and filter' refers to the ability to search opportunistically, and the 'details on demand' refers to the users ability to seek specific information within the interface. This section will investigate interaction techniques that allow the manipulation of information in order to seek interesting and pertinent results. There are four main techniques of manipulating the data to find interesting information in online interfaces: *filtering* the data according to a set of parameters, *augmenting* parts of the data to draw attention to them, *comparing* two or more different data sets, and *customizing* the display and its associated variables.

Technique : filtering

The most common method to find specific particular piece of information from a large data set is to eliminate all the data-points which are irrelevant. Filtering allows users to specify which results are not applicable to their search and remove them from the view, making it easier to spot interesting and relevant results. Filtering techniques are available in most tools that deal with large sets of data, providing users with the ability to quickly sift through the search results according to a set of specified criteria. In its most simple form, filtering techniques exist as textual commands in a search engine, used to restrict the set of search results. For example, typing in the search [africa safari -botswana] in Google, returns a set of results from the search [africa safari], but removes all results relating to [botswana]. The problem with text based filter techniques, is that they require users to learn different commands and combinations in order to make the best possible use of the tool. Furthermore, they are non-interactive. Dynamic manipulation on the other hand, can be achieved through *dynamic query tools*.

Dynamic query tools can be manipulated through either a text box (for nominal data), or a slider bar (for ordinal data). Slider bars are utilized in the Film finder and Home finder interfaces [2], both of which display data as dots on an x-y plot, and allow users to modify the dots visible through the use of slider bars. The Film finder tool displays data on different movies, plotting each movie as a symbol on the map. The symbols are positioned by the popularity of the movie (Y-axis), and its year of production (X-axis). The data can be filtered according to different variables, including title, actor and movie length. The Home finder plots the location of houses onto a map of an area. The data being displayed can be constrained by changing the range of the slider bars according to a specified variable (e.g. house cost, number of bedrooms, distance from a specified point). In search result visualization, the Grokker [39] allows users to filter their search results based on terms that they input, graying out any categories that do not contain the terms they have specified and allowing users to focus on the remaining colored categories.

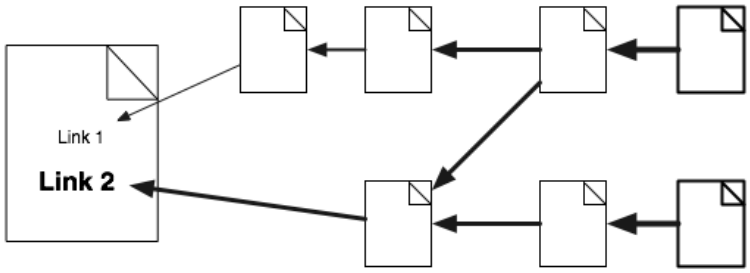

Figure 3.36: **Scent Trails (by Olston et al. [88])** modify the size of hyper link text based on the perceived relevance of the link. The relevance takes into account the distance to a relevant page (e.g. the more clicks / further away the relevant page is, the weaker the scent / less relevant the page), this relevance is amplified by several relevant search result pages intersecting.

Technique : augment

Augmenting the data is another of form of filtering, but instead of removing distracting objects from the view, it draws attention to specific parts of the data by manipulating various perceptual variables (e.g. increasing the size of an object to draw the user's attention to it). Olston and Chi's [88] 'Scent Trails', modifies the size of hyperlink text based on the relevance of that link (the larger the text, the more relevant - see Figure 3.36). More specifically, when users search a website using the Scent Trails program, the program locates relevant web pages and calculates the 'distance' (in terms of hyperlink clicks) between the relevant web page and the current web page, and this information is then encoded into hyperlinks in the current web page. For example Figure 3.36, shows two links (on the left), the text size of Link 2 (bottom) is larger than that of Link 1 (top), because the program has determined that it would take fewer hyperlink clicks to reach relevant information using Link 2, than with Link 1.

Technique : comparison

The comparison technique allows users to draw relationships between the data as well as view the data set intersections, and can be applied to both text and graphical objects. The simplest form of textual comparison is done side-by-side (placing the two pieces of text next too each other), although it is also possible to overlay documents (like laying transparencies on top of each other) or even merge documents together (similar to Microsoft Word's [90] document merging function) so that differences between the documents are placed on the same document. For textual comparison, users need to be aided in matching similar and different parts of the document e.g. the WinDiff [123] comparison tool utilizes colored highlighting to show the differences between two pieces of programming code.

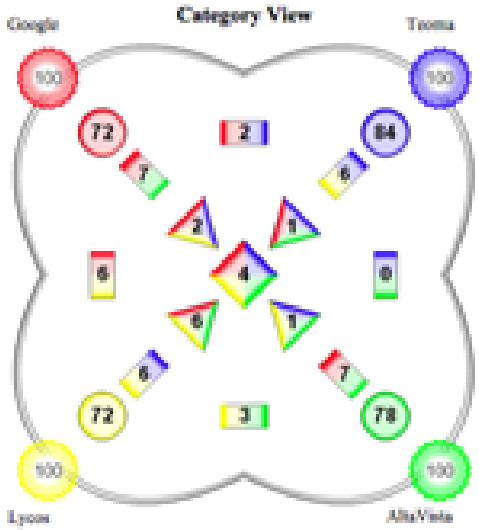

Figure 3.37: **MetaCrystal category view (by Spoerri [105]).** This shows the category view of the MetaCrystal tool. The tool visualizes the search results from four different search engines. The intersections of the sets (where a search result exists in more than one search engine) are displayed as icons of varying shapes and colours. The colours are used to show which search engines exist in the intersection (e.g. red = google.com, blue = teoma.com) and the shapes denote the number of search engines represented at each intersection (e.g. diamond shape = four searches, triangle shape = three searches). A number is written in the middle of each shape, which represents the number of search results in this intersection. In this view, the spatial location of each icon also corresponds to the sets that it belongs to e.g. the icon directly in the centre belongs contains results belonging to all four search sets.

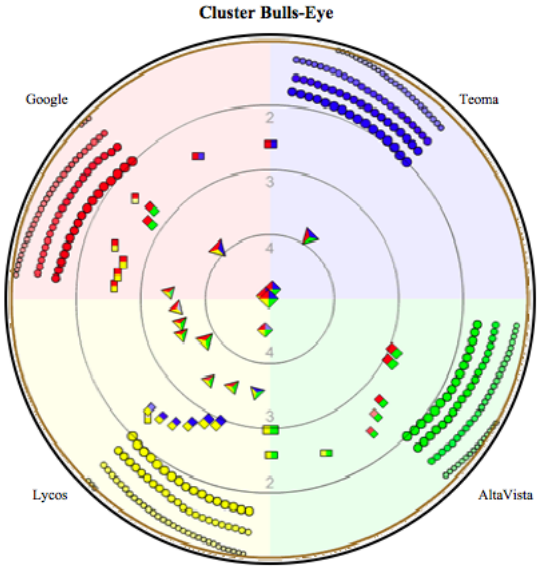

Figure 3.38: **The MetaCrystal clustered bulls-eye** (by Spoerri [105]) utilizes the same style of icons as the category view, but now there is an icon for each search result. Results with similar rankings are clustered together, and the icons proximity to the centre of the bulls-eye depends on the number of search engines that returned it.

The *Meta-search engine* compares search result data utilizing information from several different search engines in order to generate a single list of search results (examples include MetaCrawler.com, DogPile.com, and Ixquick.com). Typically a Meta-search engine will take the results from multiple search engines, combine the results (noting and filtering out duplicates) and use an algorithm to sort the results into a single rank ordered list. In appearance, the meta-search engine is identical to a normal search engine (presenting results in exactly the same way) but benefits from using several different search engines to draw from a larger pool of data. The ranking of the results is affected by the number of search engines that it appears in, and it is not uncommon for the interfaces to show (for each search result) which search engines they came from. Some Meta-search engines allow users to restrict the results shown to those results unique to particular searches.

Graphical interfaces also visualize a comparison of search result data from multiple sources. Havre et al. [42] utilizes a Bull's eye layout as part of the Sparkler tool (see Figure 3.38), which displays the results from five different search engines. The results from each of the engines is displayed as a straight line of dots emanating from a single central point (like a bulls-eye), with dots closest to the center being of the highest relevance. A brushing technique was also added to locate web pages that exist in more than one of the searches. Moving the cursor over a result which exists in more than one search engine highlights that result's position in each of the other search engines. Spoerri [105] implemented the MetaCrystal visualization, an adaption of the classic Venn diagram. MetaCrystal visualizes search result data, gathered from four different search engines, on a two-dimensional vector space. Intersections of the search result sets (web pages that exist in more than one set) are calculated and represented as icons in the graphical space. The visualization contains several different views, and the shape, positioning and color of the icons can vary depending on the view. The *Category view* (Figure 3.37) displays each intersection of four search engines, the *Cluster Bulls-Eye view* (Figure 3.38) positions icons according to their search engine, positioning the most relevant results closest to the centre, and the *Ranks spiral* (Figure 3.39) that places all the documents in a spiral, (central results being most relevant), allowing users to rapidly scan a large number of items and their titles and easily identify the top items.

Technique : customization

Customization is a combination of the discussed techniques (filter, augment, comparison), where users are given full control over what display techniques to use, what variables to show, and what data to manipulate. By allowing users to build displays tailored to their specific needs, for example, in a scatter-plot, users would be able to assign specific variables to the axes of their choice. The previously mentioned Envision tool by Nowell et al. [85] allows a great deal

of customization of its view and the variables its uses (see Figure 3.40). The tool utilizes a two-dimensional scatter-plot, where the data-points use perceptual variables (such as icon size, colour and shape) to visualize different pieces of meta-data (such as relevance ranking, and file type). However, unlike Boukhelifa et al's Multi-form glyphs tool [93] (see Figure 3.30), the Envision tool allows users to fully customize the mappings between the perceptual variables and meta-data. The axes can also be customized to show the ranges of different pieces of data.

Discussion

Filtering data in graphical interfaces through the use of dynamic query tools is an easy and efficient way for users to seek data. Both of the interaction tools (text box and slide bar) used in dynamic query tools are easy to understand and simple to use. Manipulating the slider bar gives users real-time feedback, allowing them to observe the results of various actions quickly, and thus encouraging them to test and investigate the data in different ways.

Augmenting data allows users to locate interesting results more quickly. However tool designers must be careful when integrating the technique into their tool, because the augmentation of certain perceptual variables can cause adverse side-effects which hinder user's progress. Take for example the aforementioned Scent Trails techniques [88], changing the size of the text of hyperlinks (to indicate relevance) causes the rest of the text on the page to become disproportionate and difficult to read. Additionally some hyperlinks are images and not text, so the size variable may not work. Other suggested variables to use for the visual cue are changing the text color, outlining the text and animating the text. For images, the outline border of the image could be used to indicate the links relevance. The ability to filter and augment the data of an evolving search is useful given the large amounts of information generated.

Comparison of the intersection of information from search engines,allows users to find interesting and pertinent documents, e.g. documents that appear consistently in the top 10 results of all search engines (Havre et al. [42]). Evidence shows (Lawrence and Giles [65]) that different search engines obtain their search results from different pools of information, and as a result comparison of data from multiple search engines widens the pool of information (Spoerr [105]) available to users.

Interfaces which provide the means to customize the views employed, and the data shown, allow users to develop different displays for different situations and problems. However, the increased flexibility also means increased complexity. Unlike dynamic query tools (where only the data is being modified), these tools are complicated to customize and require a certain amount of training before they can be manipulated with ease.

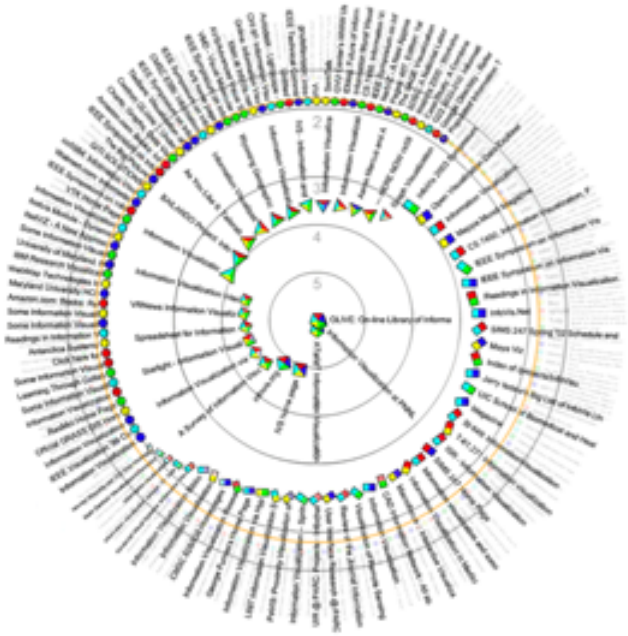

Figure 3.39: **The Rank spiral** places all the documents in a spiral. The most central documents have the highest total ranking score (averaged between the five search engines), and decrease in ranking the further out from the center you go. The view also employs a technique that pops-up details on demand.

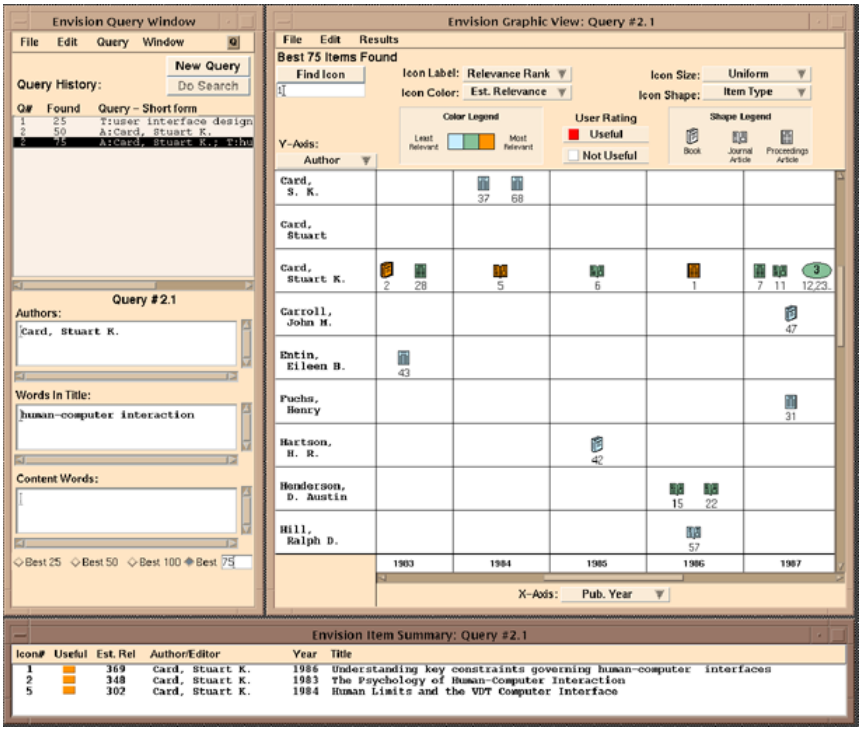

Figure 3.40: **Nowell et al's [85] Envision search tool.** This is a good example of the use of different perceptual variables to visualize different parts of the data. In the scatter-plot, search results are represented by customizable icons. The various attributes of the search result icons (such as relevance ranking, and file type) can be linked to various different perceptual variables (such as icon size, colour and shape).

3.3.2 Information Management

Evolving searches generate a large amount of information. Previous sections have investigated different ways to *visualize* this large amount of data, but the problem of how to *manage* the data has not yet been discussed. Information management is important because it helps users keep track of important threads of data, and allows users to instantly recognize and recall previously viewed data. This section will discuss the different techniques used for managing data as well as look at the different applications used for managing the data. The section concludes with a discussion on the problems with implementing Information management tools.

The two main areas where management takes place are the *data* and the *workspace*. This section will look first at the management of data, and then move on to discuss techniques for managing the workspace. When large amounts of data are generated, it is common for users to organize and group the data into 'piles' of related work (creating a structure), so that they can more easily recall the data. This form of data management is reflected in web search tools, which provide different methods of giving structure to the data. Currently web browsers provide few tools to manage information effectively, the primary tools being the bookmark tool and toolbar. Each is flawed in different ways (as discussed earlier in section - 3.1.2): the bookmark tool requiring a large amount of attention to keep organized, and the toolbar being limited in space. Users can organize data into multiple browser windows, but this is only effective in the short term and can create a large amount of visual clutter. Because web browser windows cannot be organized into specific categories, everything must be done on a window by window basis, and there is no easy way to minimize/maximize sets of web pages.

Recently tabbed browsing has been introduced to web browsers, allowing the grouping of related tabs within the same browser window, effectively allowing users to assign different windows (of tabbed pages) to different categories. Tabbed browsing is common to both Mozilla's Firefox web browser [32] and Opera's web browser [98]. Research into data management tools has concentrated on two areas, the tools that allow the *categorization* of data (manually or automatically) or introduce new interaction techniques that allow users to organize data more naturally, such as the use of *three-dimensional metaphors*.

Technique : categorization tools

Some tools allow users to place web pages within categories so as to distinguish them, and group together similar web pages. Methods vary from simply assigning specific colours to different categories to organizing them spatially into separate folders, or positions on a map. Categorization can take place either manually, with a category specified by the user, or automatically, where a category is assigned through some form of pre-assigned metric (e.g. categorization according to

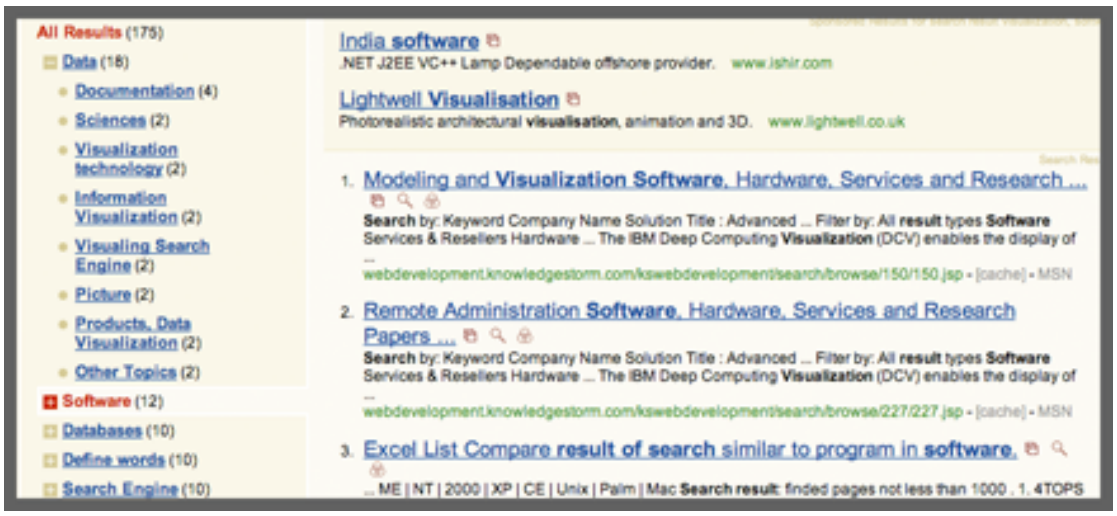

Figure 3.41: **The clusty search engine [21].** Unlike other search engines which use a ranked ordered list, Clusty organizes its search results into keyword categories (shown on the left).

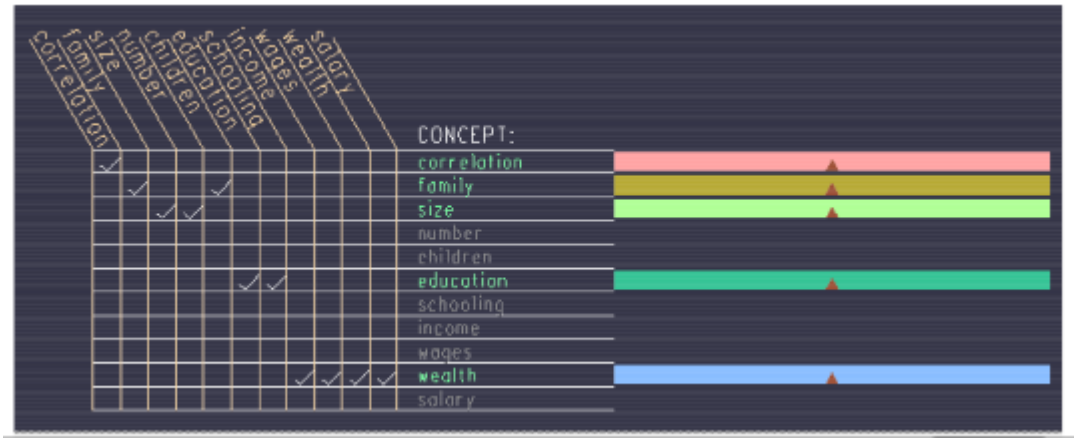

Figure 3.42: **Keyword/Concept Matrix.** Part of the NIRVE tool (developed by Cugini [24]). This can be used to generate 'concepts', which are collections of keywords. These can be used in conjunction with the NIRVE visualizations to generate clusters of data based on these concepts. When a concept is added to the matrix, users assign keywords to it by ticking on the corresponding tick boxes.

keywords). Some search engines automatically categorize user's search results according to keywords. For example the Clusty search engine (formally Vivisimo) [21], organizes its search result into categories (see Figure 3.41). The Grokker tool [39] uses a graphical interface comprised of hierarchies of circles, also categorizing its search results according to keyword. At the top level, the largest circles in Grokker form the main categories, while subsequent circles within these circles are sub-categories. These categorizations are done automatically by the search engine.

Some categorization tools allow users to affect the way that web pages are categorized, as well as generate specific categories to be used. Cugini [24] used a categorization system called keyword concepts in his NIRVE search tool. This method allows multiple keywords to be associated with a single 'concept' (which is in turn represented by a single word), for example, the concept 'wealth', would include the keywords 'salary', 'wages' and 'income' (as well as the keyword 'wealth'). These keyword-concepts could then be used in conjunction with NIRVE's three-dimensional search result visualizations (such as the aforementioned three-dimensional scatter-plot) to generate clusters of search results. Users can generate their own concepts, simply by adding a new concept to the keyword/concept matrix (see Figure 3.42) and ticking the boxes of keywords to be associated with the concept.

The Scatter-gather tool by Hearst et al. [44] (see Figure 3.43) was developed as an alternative to ranked search lists. When scatter-gather retrieved a set of search results it would automatically categorize the results into a set of clusters based on keyword categories. However, scatter-gather introduces an extra step in the search process, where feedback from users (selecting interesting clusters) affects the next set of results returned. When this feedback is sent into the system, the unwanted clusters are discarded and new, more specific, clusters are generated and returned to the user. In essence, users begin with a set of very general categories (some of which may contain many different subjects) which eventually, through the process of feedback and re-clustering, narrow down into the specific clusters that users are interested in.

The Aspect windows tool (developed by Swan et al. [112]) is another data management tool that combines automatic and manual processes. Aspect windows are integrated into the AspInquery information retrieval system (see Figure 3.44). When users find a document to retain and categorize, the document can be dragged from the ranked list into the aspect window's document list, where a new 'aspect' (category) will be generated automatically. When aspects are initially created, they are analyzed and the five phrases which best describe the document (taken from the text via statistical analysis) are assigned to the aspect. Users can then manually modify and assign keywords or labels to the aspects, as necessary.

Figure 3.43: **The scatter gather tool (by Hearst et al. [44]).** This search tool organizes search results into clusters based on keywords. The tool also has a interesting method of feedback, which allows users to select which clusters are relevant to the search, and which are not. This feedback is returned to the system, which removes the irrelevant clusters, and defines new clusters based on the data that the users specify as interesting. Theoretically this should filter out irrelevant search results and fine tune the search result set.

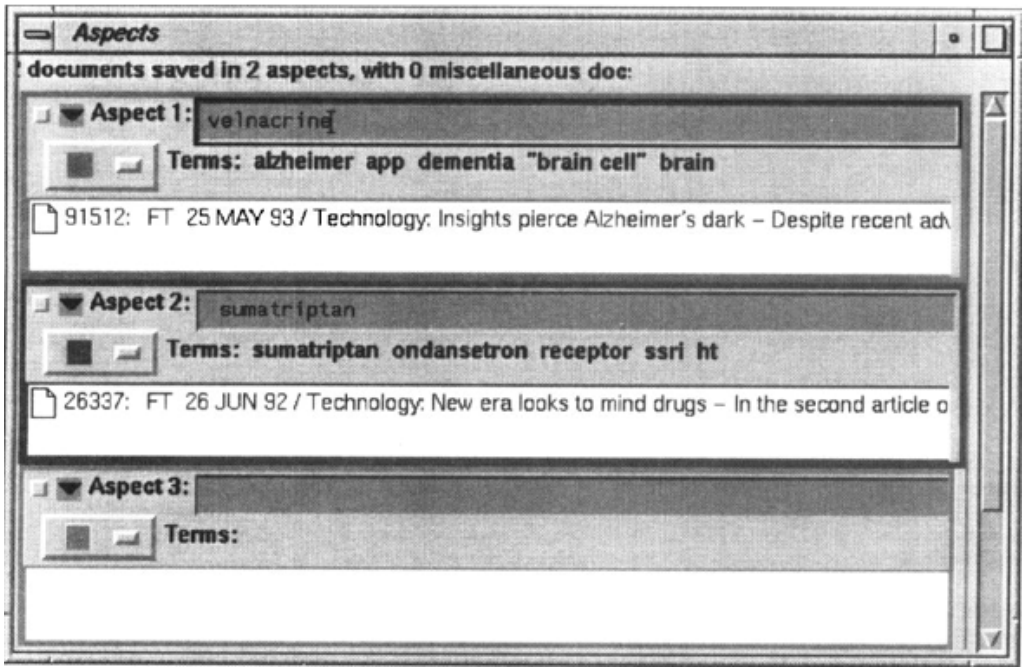

Figure 3.44: **The aspect windows interface (by Swan et al. [113]).** This tool is a support tool for the AspInquery information retrieval tool. It allows users to drag results from the main view, into the aspect window view, where the result is assigned to an 'aspect' (category). The name, labels and colour of the aspect can all be assigned by the user.

Technique : three-dimensional metaphors

People organize data in different ways, whether it be organizing books on a shelf or in a pile on the floor. Techniques which manage data through the use of three-dimensional metaphors can mimic real life organization and categorization systems. For example, the BumpTop desktop (by Agarawala et al. [1]) metaphor is a three-dimensional representation of a physical desktop (see Figure 3.45) which mimics real-life physics, allowing objects on the desktop to be 'picked up' and 'thrown' across the desktop, as well as placed into 'stacks' and 'piles', much the same way as would be done in real life. This form of management has the advantage of being completely natural to the user, and can be adopted with great ease.

The WebForager workspace (developed by Card et al. [18]) also recreates a physical desktop, but unlike BumpTop does not attempt to mimic the interaction physics of real-life. Instead it concentrates on incorporating several data structures that users are innately familiar with and can understand easily. For example, information is stored as 'books', and several books can be placed and ordered on a 'book shelf' (represented as a three-dimensional book and bookshelf respectively). The advantage is that users can draw inferences from the metaphors, e.g. users know that there will be textual information within the books (by drawing comparisons to real-life books), and know that books can be placed in order on the bookshelf.

Figure 3.45: **The BumpTop desktop metaphor (by Agarawala et al. [1]).** This shows the three-dimensional desktop tool, BumpTop. The square tiles on the desktop represent files of different types. The physics of the desktop allow the files to be 'picked up' and 'thrown' around (or into corners) as if in real life. Files can also be 'dropped' on top of each other form 'piles' (like in real-life). This form of data management is easy to learn and use because of its inferences to real-life interactions.

Workspace management

Management of the workspace is important for keeping the user's interactions efficient, and includes making sure that frequently used tools/data are close to hand, and in view at all times. The majority of web browsing tools use a fixed structure, limiting the customizability of the layout of the interface. By contrast, operating systems (such as Microsoft Windows [90] and Apple OS-X [89]) utilize a 'window' and 'desktop' system, which allow a high level of customizability and workspace managment. Windows can be moved freely around the desktop, as well as resized and hidden, allowing maximum flexibility in terms of which views are available and in which positions. Some Linux based operating systems allow users control over several desktops which can be switched between.

Some tools mimic this desktop interface (e.g. Corel's Paintshop Pro [126]), effectively generating a mini-desktop within the tool window. Inside the mini-desktop, the different tools and functions are represented by floating windows which can be resized and moved around as if on a normal desktop. Recently there has been a move towards 'open workspace' tools (e.g. Adobe Photoshop - OS-X edition [52]) in which the tools of the program are not tied together and are each treated as separate windows on the desktop. The management of a tools workspace allows users to create a more effective work environment. An example of this, is the Snap-together coordinated visualization (by North and Shneiderman [83]). The tool links together several windows, each containing a different visualization tool. Each of the tools is coordinated, and draws from the same data. The tool windows can be moved around and re-arranged so that they suit the user's needs more closely.

Discussion

Tabbed browsing is an effective method of keeping web pages organized together, however arranging tabs into specific windows to generate specific categories requires manual organization and effort. Moving tabs (along with their entire web page histories) between web browsers is not trivial, and identifying tabs can also be tricky if there are too many of them. Categorization tools allow users to organize their search results, but tool designers must consider the amount of automation used in the tool. For example, increased automation is useful when generating a large amount of web pages from different subjects, however this also increases the chances of the system placing web pages into erroneous categories. Decrease in automation means more accurate categories, but more work for the user. The ability to manage the workspace is useful but can sometimes be troublesome for new users who are unfamiliar with the tool.

3.3.3 Overview

This chapter has looked at the five interface issues related to the evolving search addressed in this work, and presented different technologies and techniques that solve these issues. These were related to (1) opportunistic searching, (2) information recall, (3) information visualization, (4) information seeking and (5) information management.

Section 3.2.1 identified the inadequacies of current search interfaces for browsing opportunistically. The textual list format employed by current search interfaces uses a linear '10 results at a time' representation, which forces users to press the 'next page' button each time to progress. Focus-and-context, and graphical display techniques were both suggested as solutions to this problem, but each have their own disadvantages. Focus-and-context can distort the data to the point that it becomes unreadable (as illustrated with the FishNet browser [7]), and graphical display techniques remove much of the textual context.

The different automatic and manual history systems were discussed in section 3.2.2, and a set of criteria for the design of history systems (based on Kaasten et al's [61] list) was presented. It was noted that a system which combined both the automated and manual aspects of history systems would be ideal.

Section 3.2.3 discussed the use of graphical display techniques for visualizing large amounts of data, and different two-dimensional and three-dimensional techniques were presented (such as the scatter-plot, spiral and topographical map techniques). It was noted that for graphical displays, it is important to use the correct combination of interface and perceptual variables, that are understandable to users.

Techniques for manipulating the information displayed to seek interesting search results were discussed in section 3.3.1, including techniques to limit the display to relevant results (filter), increase the visibility of interesting results (augment), view similar information between data sets (comparison) and customize the views of users. Filtering and augmenting techniques were identified as simple to use, and provided immediate feedback to changes made. Comparing the data between different search engines was seen as a viable way to find new information, and the idea of viewing the intersections of multiple searches was touched upon.

Techniques for managing the large amounts of data generated by evolving searches was discussed in section 3.3.2. The use of categorization tools was looked at, specifically the differences between automatic and manual categorization systems. It was determined that a management system that provided elements of both automatic and manual categorization systems was ideal, and that currently, no systems implement these together effectively.

Chapter 4

Design

Chapter one listed the six different evolving search interface issues that were reported by Bates [6] in her research. She believed that these issues needed to be addressed in order to build a more effective interface for performing evolving searches. Three of the six interface issues she identified,are addressed in this work, these are the issues of (1) opportunistic searching, (2) information visualization and (3) information recall. Bates also discussed other interface issues related to (4) search strategies, (5) analogies, and (6) information switching, which are not addressed in this work due to constraints in technology, time, mappings and interface conflicts. Each of these is discussed in turn.

Many of the strategies (such as area scanning, author search and journal run) were all directly associated with actions and structures present in a physical library that are not present in the online environment. For example, users cannot 'area scan' to find websites that are physically located nearby to each other. Other strategies mentioned have already been implemented, for example, backwards chaining (referring to following references in a book, onto other sources) is already reflected in the hyperlink structure of the World Wide Web, where users can follow information sources backwards through the simple click of a link. In short, the issue of providing for different search strategies was not addressed, due to mapping constraints; some of the strategies could not easily be mapped to existing search strategies because of differences between the information structures of the library and the WWW and some of the strategies had already been mapped.

Bates suggested that modeling the search interface after a real-life analogy would facilitate user's understanding of the interface and how to interact with it. However, adhering to a specific analogy reduces the flexibility of a design, meaning that certain combinations of analogy designs and visualization techniques are not compatible. For example, a three-dimensional book analogy allows users to browse textual information quickly (by 'flipping' through the pages), however it

removes all structural content. For example, web pages that are normally arranged in a tree hierarchy, are presented instead in a linear order such as the pages in a book. This means that users do not benefit from an overview of the hyperlink structure, nor can they opportunistically traverse the hyperlink structure in a non-linear fashion. The design for an evolving search interface was not based around a real-life analogy because of potential interface conflicts. By not using an analogy, the design was less restricted in terms of techniques which could be applied.

Bates stated that, when searching, users access information from different sources, and hence would benefit from being able to switch between different sources quickly. This issue was common back in the days when switching information sources in a library meant physically relocating oneself, and switching between different databases could not be carried out with the single click of a button. Nowadays this is less of an issue, since the WWW seamlessly merges together several different information sources, and the issue could instead be construed as the problem of accessing and switching between different search engines. However, implementing a tool that could access several different search engines is difficult, since at the time of design, very few search engine companies provided access to their Web APIs. The inability to provide access to multiple information sources for information switching was due to technological constraints.

In addition to the constraints mentioned above, one other primary constraint was time. While there are many different technologies available for designing web search tools, many require extensive knowledge in various different programming languages. Given the limited time to complete this work, it was decided to concentrate primarily on providing services available in the Java programming environment. This chapter will discuss the different tool designs that helped shape the development of the evolving search interface. The final implementation of the evolving search interface,was based on a set of prototype tools which were developed with the specific purpose of solving evolving search interface issues. Specifically, this chapter will look at the design and implementation of *the visual-bracketing tool*, *the comparison tool*, *the visual-history tool* and *information management tools*.

4.1 The visual-bracketing tool

The traditional search result representation utilizes a textual format, which is not ideal given the massive amount of screen space that textual representations utilize. As a result, most search engines break up their search result lists into pages of ten results. Such a discretized representation makes it difficult to search the data opportunistically as users are forced to view the results a page at a time. Graphical interfaces provide methods of displaying many more results in the same screen space, and provide the required opportunistic search strategy; however, users will be faced with new symbols and icons whose functions are not necessarily implicit. Even

if a more easily recognizable format was presented (such as a simple two-dimensional scatterplot) it would still take users time to develop an understanding of how to use the system (e.g. what do each of the axes represent? what do the icons represent?) in addition to becoming accustomed to searching with it.

Another technique that could be used to display the large amounts of data is the focus-and-context technique. Previous examples of focus-and-context ([49], [67], [69]) have dealt with text by distorting it or representing it at a lower level of detail, often making the text barely readable. Instead, a design was chosen which would represent the text at a different semantic level, while still giving meaning to users. For example, reducing a search results information, from a title, snippet and URL, to simply the title or url. This was the concept behind the *visual-bracketing* design, which allows users to browse search results *opportunistically* by removing the traditional 'page-at-a-time' representation, and visualize large amounts of search result information effectively through use of a focus-and-context technique. This section describes the design of the visual-bracketing technique, as well as two implementations that utilized the design, the visual-bracketing tool, and the coordinated bracketing tool.

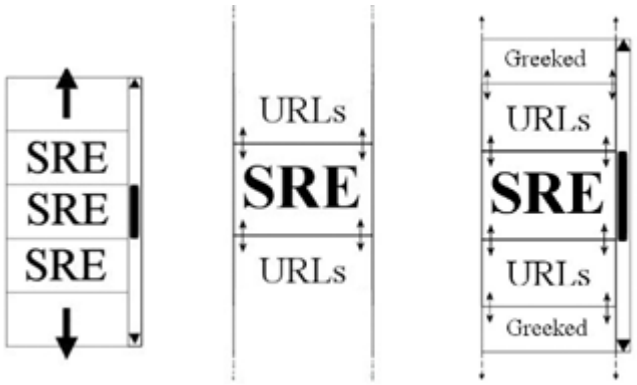

Figure 4.46: **The visual-bracketing design.** Schematics showing bracketed visual depictions of the Search Result Elements (SRE). Left, shows the traditional scroll list, each result in full detail. Center, shows the bracketed version with two levels of detail: full detail SRE that is bracketed by a simplified view of URLs. Right, shows a three-level bracketed view, with the third level being depicted by greeking (see work by Robertson [96].)

4.1.1 Design

Like all other focus-and-context designs, visual-bracketing allows users to focus on interesting parts of the data while still viewing the surrounding context. The idea of visual-bracketing is inspired from bracketing in photography, where professional photographers "take a series of images of the same scene at a variety of different exposures that bracket the metered (or manual) exposure" (see [15]). The visual-bracketing effect is achieved by displaying different semantic information in the fore and after visualizations. The inner part contains the detail view while

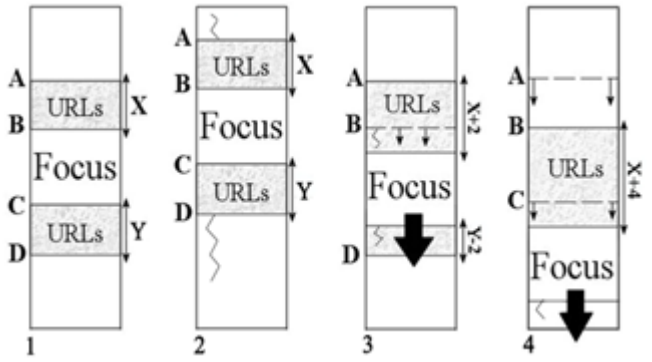

Figure 4.47: **Visual-bracketing : scrolling mechanisms** Schematic showing the different scrolling mechanisms. (1) The original bracketed view. (2) Constant width method where the surrounding brackets X,Y are kept constant and the areas above and below A,B respectively are enlarged/reduced. (2) and (3) show the inner-bracket scrolling method, where the panes may scroll inside the bracket, up to the bracket end (3) or push past the bracket (4).

the bracketed visualizations contain the context information at a lower level of detail. In the case of web search result visualization the inner part contains full information about the search result, while the bracketed views contain less detailed information such as only the URL (see Figure 4.46 - center). Indeed, further bracketed views may enfold the former, showing less detail still. Thus the information is depicted in different views using various levels of detail (see Figure 4.46 - right).

The different level-of-details (LOD) may be generated in a variety of ways, for example, abstracting the information in some way or simply excluding information. In the design, the second level depicts less information while greeked lines represent the third LOD. *Greeking*, as used by Robertson [96] as part of the document lens visualization and SeeSoft [29] in software visualization, exchanges the characters of the text with straight lines. This technique is useful to provide summary and overview information about a text document.

A sliding window methodology is also applied to the design, such that users may directly slide the focus view up and down to change the information that is displayed in the center (and corresponding bracketed) views. This acts in a similar way to a fish-eye view [34], where the center window displays the full resolution and a lower level-of-detail is shown either side. But with this method there is a non-continuous change in the level of detail rather than a gradual change (along with a coincident semantic change). Moreover, this visual-bracketing concept is similar to the aforementioned perspective wall design [69] (see chapter 3), albeit horizontal rather than vertical. The perspective wall technique displays the data in three dimensions on a two-dimensional wall. Users focus on the information in the center wall, with the side parts of the walls displaying the context; information on the wall may be scrolled to change the focus. In the visual-bracketing design, bracketing may occur at multiple levels (brackets of brackets), a semantic level change is employed and users may alter the bracket's size.

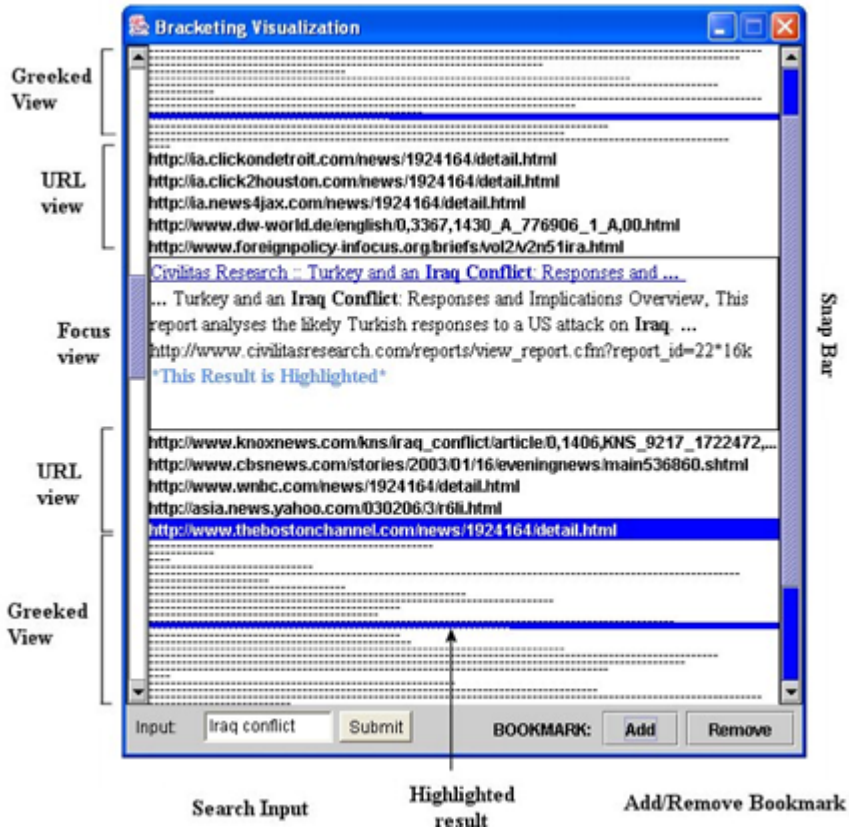

Figure 4.48: **Visual-bracketing tool.** Screen shot of the bracketing visualization. The focus view shows the full detail of the Search Result Element, solely the urls are depicted in the adjacent bracket, finally greeked text (of the urls) are shown in the outer most level.

Different interaction methodology were considered, that would allow users to manipulate the amount of information within the bracket. Two designs for interacting with the focus window were considered. In the first design, the width of each window bracket is kept constant, apart from the outer most panes, as depicted in Figure 4.47 (left); thus, users move each of the brackets together as one unit, they are locked together (as demonstrated in Figure 4.47 : centre-left) and the information changes appropriately. In the second design, (Figure 4.47 : centre-right) users may grab each inner bracket (the focus view) and move that up and down within the restriction of the width of the encapsulating bracket. If users attempt to push past the boundary (Figure 4.47 : far right) of the next bracket then that bracket width changes and moves as well, etc.

4.1.2 Implementation I : visual-bracketing tool

The *visual-bracketing tool* (see Figure 4.48) was implemented using the aforementioned bracketing model. The tool was implemented in Java 1.4 with Swing components, and utilized the Google Web API to retrieve its search results. Tool interaction begins with users typing and submitting a query to the tool, which in turn retrieves a set of results (via the Google web API).

These results are then visualized according to the bracketing concept; one result is shown in full detail (focus view), the next 5 results are displayed either side as URLs with the remaining results displayed as greeked lines. Users can scroll up and down to alter the information in the focus, or click on any result (greeked or URL) to change to a new focus view; the rest of the visualization rearranges itself accordingly. Users can view the associated web page by clicking on the link in the focus view, which opens the page in a new browser window. Users can also bookmark results. These bookmarks are highlighted and allow them to return to view these results at a later stage. When two or more results are bookmarked, another scrollbar appears on the right-hand side (named the snap-bar). The snap-bar allows users to move (snap) between highlighted results, enabling them to quickly return to marked elements. The lines in the greeked view represented the size of the respective web pages, but could have been used to represent other types of metadata.

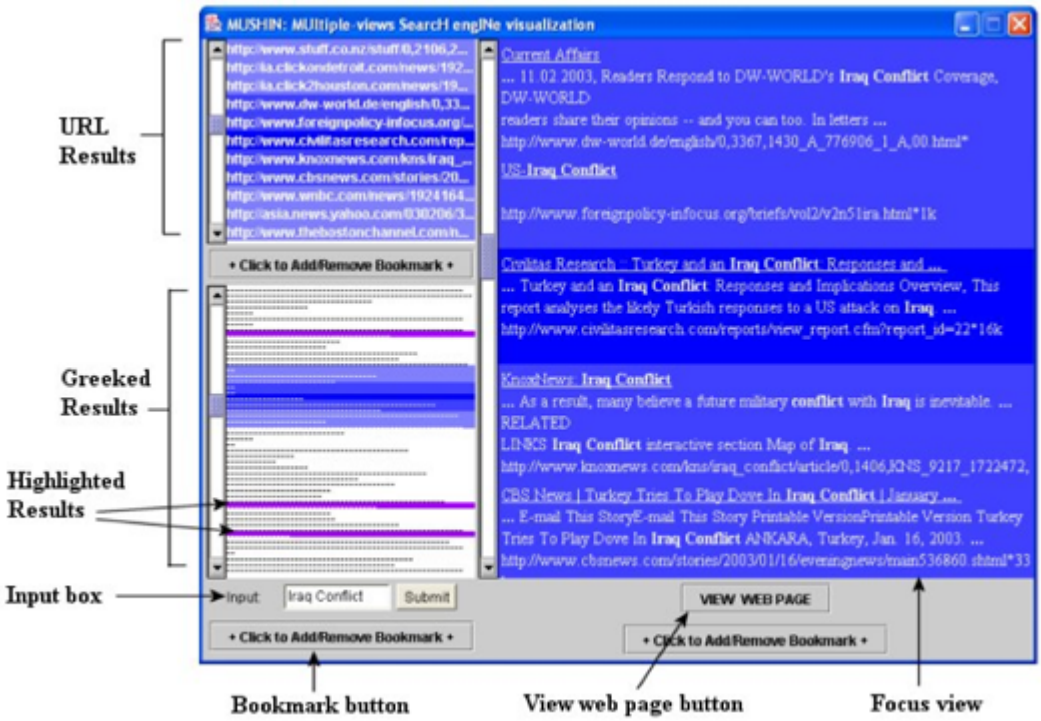

Figure 4.49: **Coordinated bracketing tool.** The multiple view variation of the visual-bracketing tool. Separate semantic information is shown in each views: greeked lines, URLs, and full SRE information, respectively. Different shades show the bracket information, and help to coordinate between the different views.

4.1.3 Implementation II: coordinated bracketing tool

The *Coordinated bracketing tool*, a more advanced model of the visual-bracketing tool, was also developed. The tool separates the semantic information into multiple views; a screen shot of this

implementation is shown in Figure 4.49. Each view displays (in their own view pane) different semantic levels of the same search result element (SRE): full SRE information, URLs, greeked lines, respectively. The pane containing the greeked list displays the entire list, the URL list displays ten results at a time, and the fully detailed pane displays five results at a time. An element of coordination among the views was added, such that when the selected SRE changes in any of the views, the other views automatically update, keeping the views consistent. Again users may bookmark elements, and bookmarking a result in one view will highlight it in all the other views.

The use of coloured bands within the visualization allows users to match the currently viewed data elements in one viewing pane with those in the other view panes. The highlighting colour is different so that users can easily distinguish which results have been bookmarked. As with the visual-bracketing tool, the length of the greeked lines visualizes the size of the document. This provides an additional layer of information and gives users an additional visual cue to use when searching the list of results.

4.1.4 Evaluation

A preliminary user trial was performed with both implementations. Six users were given a brief demonstration of the program and then encouraged to interact with the visualization. Afterwards users were asked to complete a questionnaire; the findings have been encouraging. Users found the controls of both models easy to understand and manipulate, and favored the coordinated bracketing tool stating that it made the search results easier to view and provided extra functionality compared to a normal search engine (e.g. Google). The bookmark function was praised most as a useful tool to identify and remember SREs of interest. Users also agreed that the greeked visualization provided a useful overview of the data (through the size representations in the greeked lines) and that it aided them in their search by providing additional visual landmarks.

The designs of the visual-bracketing tool and the coordinated bracketing tool formed the basis of the future design of the EvoBerry tool. The visual-bracketing technique provided a simple method of visualizing large amounts of search results in an understandable format, addressing the issue of *information visualization*. The technique also displayed the data in such a way that users could jump between different results in the list quickly, addressing the issue of *opportunistic searching*.

4.2 The comparison tool

One of the key features of the berry-picking model is the generation of multiple searches, and the fact that users absorb information as they move from search to search. However, users can become overwhelmed by the mass of information presented, making it difficult to seek interesting information. One suggested method of finding information was the use of *Comparison views*, comparing the intersections of web searches generated throughout the search session. A comparison view can pre-attentively visualize the associations between the multiple datasets by explicitly demonstrating and annotating these features. Through such comparisons users can gain a better understanding of the searched information and they can more effectively browse, because their attention is drawn towards significant features in the results. As these observations are explicitly represented, the cognitive overhead of switching from one result-set to another is reduced.

4.2.1 Design

The data to form the comparison can come from various sources. Users may wish to compare the results from a range of search engines, to observe the popularity of different sites or to find particular results that one search engine may not display. Alternatively, users may wish to compare the results from multiple searches using different keywords, to find links between different subjects or to help users consolidate towards some ideal search terms. This may involve the use of intersection (A $\cap$ B), union (A $\cup$ B) and difference (A-B) operations and meta or statistical information may be compared.

There are different ways to compare data, one of the earliest text-based comparison tools could be the English Hexapla New Testament [9]. This was printed in 1841 and displays the original Greek at the top of a page with six different English translations underneath in parallel columns (three on the left page and three on the right). This demonstrates the simplest form of comparison, side-by-side comparison which can be further extended to utilize annotations (or various cues) between the parallel views. For example, similar information can be concurrently highlighted (in the same color) or connected together with circles and arrows (see Figure 4.50).

Other methods of comparison exist, for example, overlaying multiple visualizations will generate a single layered view in much the same way as overlaying a series of overhead transparency foils would generate a single view. This enables a visual comparison of the multiple views and does not alter the format of either visualization. Various parts of the information could be shaded out, to bring the similarities between data sets to the foreground. A merged view shows multiple datasets in one view, where each of the differences are shown in context with surrounding information. For example, Microsoft Word provides a merge document facility that

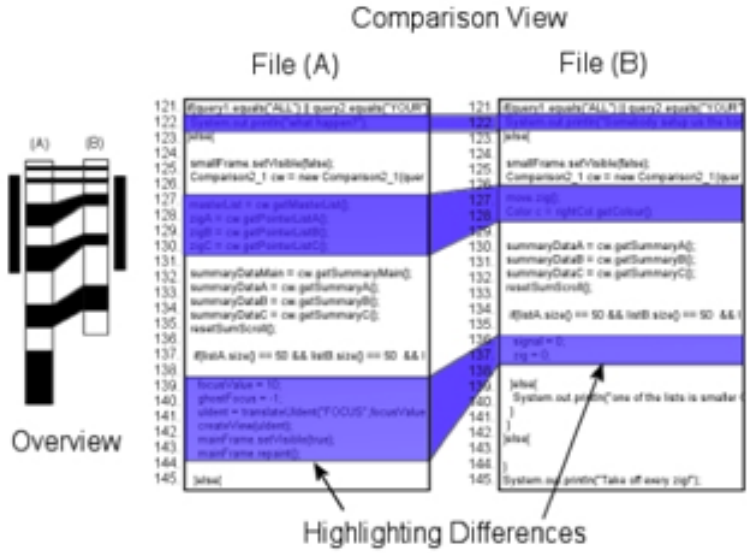

Figure 4.50: **Model of Code Comparison.** Two files (A & B) are compared, differences are highlighted in color and an overview of the two documents is shown on the left. Colour highlighting is used to link similar results (Similarity principle) and synchronized scrolling is used to align the two files while they are being viewed (Symmetry principle).

aggregates different versions of the same document together. The sentence and word differences are illustrated by colour highlighting and word strike-throughs. Users are given the responsibility to accept or reject individual changes.

The abstract comparison of data differs greatly from the aforementioned methods. This type of comparison moves away from traditional textual representation and moves towards a more graphically centered approach, where data is structured into a visual encoding. Havre et al [42] utilize abstract views in the SPARKLER visualization; in their tool a dot represents a search result, a group of dots represent a list of search results and multiple groups of dots represent several searches.

In order to compare information from different sources, users must be able to visualize the various associations between the different elements. This can be achieved through the use of the Gestalt principles [119], a set of theories of perceptual organization. The principles impact upon comparison visualizations in different ways, for example, coordinated multiple views [94] link the current item in focus across several different views, so that changes in one view will affect all other views similarly (the gestalt principle of chronology). This is apparent in both the Vdiff tool [5], and North and Shneiderman's Snap-together tool [83]. Coloured highlighting links together similar objects in different sets of data, so that the items are understood as belonging together (the gestalt principle of similarity). This is utilized in the SPARKLER visualization by Havre et al. [42].

Figure 4.51: **The SES (search engine similarity) tool.** This contains four views, the summary view (top left), the overview (bottom left), and bracketed view (right). Users input queries in the bar along the bottom.

4.2.2 Implementation

The aim was to design a tool that allowed users to identify results that were both similar and interesting, through a process of comparison. Two criteria were stipulated, (1) the design should link similar results between different views, so that the information appears coordinated, and (2) the design should provide a summary view of the data which displays an intersection of results from different search terms. The comparison tool was designed and implemented as part of the SES search engine similarity tool (see Figure 4.51) a tool for visualizing the intersections of multiple searches. This section details how users operate the tool, then discusses the three sub-components of the GUI: the summary view, overview and bracketed view, and detail the use of coordination between the views.

Users begin their interaction by inputting two or three sets of search terms related to the same topic. These terms are then submitted to Google using the Google Web API [3]. Two (or three sets) of search result elements (SREs) are returned respectively. Each SRE represent an individual url, and contains information concerning four variables: title, URL, paragraph of text and page size. The information is then visualized in three coordinated views: summary, overview and bracketed view.

The summary view (top left of Figure 4.51) displays only the websites that appear in two or more search-result lists. This view consists of a tabbed pane, with the main tab showing a

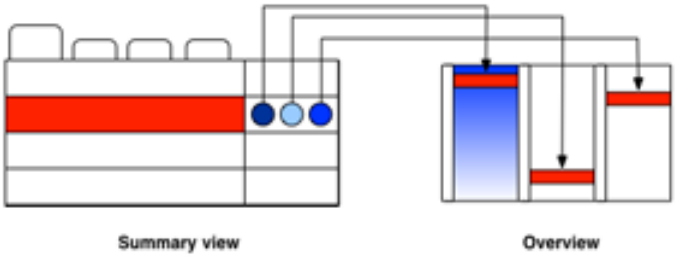

Figure 4.52: **SES : Circle Glyphs.** Each of the circles are shaded in a value equivalent to their ranking (a darker shade of blue represents a higher ranking).

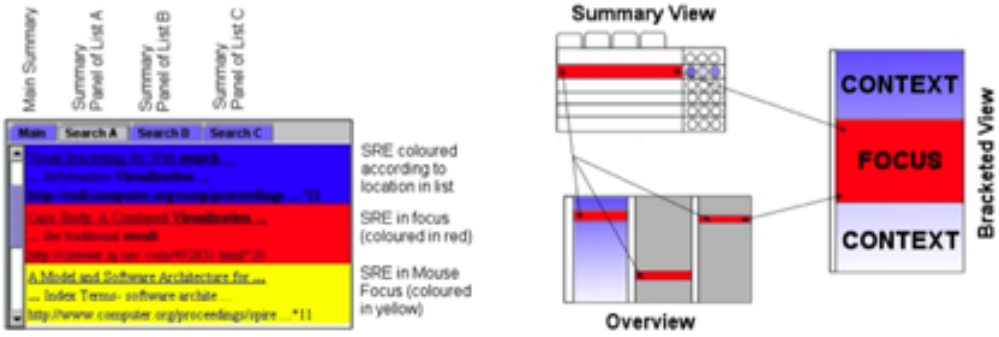

Figure 4.53: **SES : Summary view (left).** Each tab contains a detailed view of each specific search, including page title, fragment of the text, URL and page size. **Coordinated views (right).** When users select a result in one view, all the other views are updated and the result is highlighted in the other views.

summary view, that contains a list of URLs each with three circle-glyphs A/B/C. The circles are shaded in a value equivalent to their ranking (a darker shade of blue represents a higher ranking, see Figure 4.52), unless the particular result is not present in that search result, in which case it is coloured grey. The results from each search (A, B or C) are aligned in columns. Each of the other three tabbed lists filters the data in the main summary view according to search, and shows detailed information for each specific search, including page title, snippet of the text, URL and page size.

The overview panel (bottom left of Figure 4.51) provides a view of the whole data set. Comparing lists of search results in context allows users to identify interesting results in both the immediate surroundings and within the macro view of the list. This is achieved using greeking. In this view, the length of the lines represent the page size and the parallel columns each represents a search result list. The information within each column can be ordered in several ways, such as, by page size or rank order. In the implementation, each of the columns are ordered by rank. When one column is selected (e.g. column A) this becomes highlighted and the intersections with the other lists (columns B and C) are shown in a shade equivalent to the rank of the result in the selected overview (column A). **The bracketed view**, depicted

on the right of Figure 4.51, is based on the visual-bracketing principles mentioned previously and contains a detailed view of the currently selected list in the overview panel. As in previous designs, users can open a webpage by clicking on the title of the result.

Coordination is an important part of comparison visualization. For example when viewing a result in one search result list, users need to be aware of the location of similar occurrences of that result in other lists. When users select the focus of the summary view (bottom left) the bracketing view updates to depict the selected information, see Figure 4.53 (right). Second, a form of dynamic brushing is used to highlight every URL that has the same hostname (highlighted in yellow). As users move the cursor and brush over an SRE, every other occurrence of that SRE is highlighted, wherever it is located. Finally, every occurrence of the hostname that is focused in the bracketed view is highlighted (in red) in every other view. The dynamic brushing, referred to in this work as the *Similarity highlighting*, helps users to find similar (and thus interesting) results quickly.

The Program was implemented using Java 1.4. Data was obtained through the Java bindings to the Google Web API [3]. The program uses a model-view-controller design pattern where the results are stored in a central data-model and the windows represent views on this data. Manipulating either the mouse or the scrollbars will trigger a set of listener objects which will manipulate the visualization accordingly. The listeners also control the colour-shading, which highlights results as the mouse cursor moves over them.

4.2.3 Evaluation

In order to test the effectiveness of the SES tool, it was compared to a standard ranked ordered list visualization (Google). The original hypothesis was that users would benefit from a tool that explicitly displayed the similarities between multiple sets of results, and as a result would be more efficient at choosing relevant results. One of the foremost concerns with this hypothesis was that there was no way of testing the direct impact of the display techniques, since a search is tempered by the information users acquire from browsing search results pages. To effectively test this hypothesis all other elements in the environment had to be controlled, which meant that certain steps in the information seeking process had to be carefully choreographed so as to eliminate any influences from these phases. Consequently, users were given pre-determined search terms, were restricted from viewing webpages, and restricted from using advanced search options, forcing users to judge the relevance of search results by the display techniques alone.

Experimental design

The information retrieval task given to users was presented in the form of a question. Appropriate questions had to be chosen carefully, as was shown by Dempsey et al. [27], who illustrated

how difficult it was to find a neutral question when designing a study. They discovered that certain key variables (such as proper names) were excellent discriminators and greatly affected the results.

The task chosen for the evaluation, was to *"List the names and depths of the Worlds Deepest Seas"*. The terms given were (1) [Worlds Deepest Oceans], (2) [Seas Deepest Depths and Oceans], and (3) [What are the Worlds Deepest Seas]. The test subjects comprised of a mixed gender group of 20 people from various backgrounds. At least half of the group were non-computer professionals but all had a working knowledge of search engines. One half of the group (the control group) used Google, the other half (the test group) were given the SES tool. Each subject was monitored in three ways, through their interactions with the keyboard and mouse, through the data submitted via the user interface, and by human observation. Monitoring software was used to capture screen shots of interactions on the screen, using the mouse and keyboard. Each of these capture sessions was stored as both a set of thumbnails and an animation. All interactions with the WWW were passed through an Apache server which had been setup with the proxy module enabled, ensuring that all requests were filtered through a script that both controlled and automatically monitored the information returned.

Before starting each test, users were given a demonstration of how they were to conduct their tasks. They were also given an opportunity to familiarize themselves with the controls of their interface (the Google search engine for the control group, and the SES tool for the test group). A webpage interface was used to guide users step-by-step through the process. Users in both the control and test groups were given a search scenario and three sets of search terms (as stated previously). In the *control study* each of the sets of search terms were represented as a hyperlink which opened up a modified Google web page (via the proxy), where links to all of Googles facilities (e.g. image search, news search) outside of its web search were disabled. Users were then told to browse the results and select what they thought were the five most relevant results to the search. To mark a result as relevant, users clicked on the title of the result, which would open up a relevance rating page (as stated previously, users were deliberately restricted from browsing web pages). On the relevance rating page, users could rate how relevant they thought the result was, based on the meta-information alone and compared to the scenario given. The results were rated from one to three, one being least relevant (see table 4.9). Users were required to locate five relevant results for each of the three sets of search terms. In the *test study*, the three sets of search terms were entered into SES and the comparison between the result sets was displayed. As with the control group, the test group users had to select five results which they deemed relevant to the scenario, and rate each of these. Finally users filled in a short questionnaire at the end of the test.

$$\frac{X_t - X_c}{\sqrt{\frac{\sigma^2{}_t}{N_t} + \frac{\sigma^2{}_c}{N_c}}} = \frac{\textit{Difference between group means}}{\textit{Variability of groups}}$$

X_t = Mean of the test group
X_c = Mean of the control group
$\sigma^2{}_t$ = Standard Deviation of test group
$\sigma^2{}_c$ = Standard Deviation of Control Group
N_t = Number of users in the test group
N_c = Number of users in the control group

Figure 4.54: **The t-test Equation**

Rating	Criteria
1	The Web page doesn't contain the data you are seeking, but the information can be found on another page in the website, or via a link on the website.
2	The Web page contains some of the data required but not all
3	The Web page contains all the relevant data

Table 4.9: **Criteria for selecting relevant web sites.** When selecting a result as relevant, users must assign a value as well, for its percieved relevance with regards to the query.

		Not rated	User's ratings		
		0	1	2	3
	1	0pts	3pts	2pts	1pt
Expert	2	0pts	2pts	3pts	2pts
Ratings	3	0pts	1pt	2pts	3pts

Table 4.10: If users selected a relevant result, they were assigned points based on what the perceived relevance of the result was, compared to the relevance rating assigned by a group of experts. If users selected an irrelevant result, zero points were recieved.

	Pages of web results		
	Page 1	Page 2	Page 3+
Search 1	46%	44%	10%
Search 2	28%	30%	42%
Search 3	60%	16%	2%

Table 4.11: This table shows the number of pages investigated by the control group (using IE) over the three searches (shown as a percentage of the total number of pages investigated).

Results

The results were analyzed using two weighting strategies. First, users ratings were compared with those determined by a small group of information-seeking experts (R1) and weightings were assigned proportionally to the difference between the two results (table 4.10). Second, users were allocated a point for each relevant result they selected, irrespective of any rating given (R2). A statistical analysis of the results was performed, using *Welch's t-test* [26]; this method assesses whether the means of two data sets are statistically different from each other. This is important because a comparison based on averages alone does not take into account the spread or variability of the scores. For example the difference between two means may be the same for two experiments, but one may have a high distribution across the data set, and the other a lower distribution. The higher distribution will hence have a larger overlap than the lower distribution, and therefore will be a more similar set of results (and hence less significant) than the lower distribution data.

Once a result has been obtained from the t-test formula (see figure 4.54), the number coupled with an alpha/risk level, and degree of freedom is compared against a table of results. The Alpha level is normally set to 0.05 (meaning there is less than a 5% possibility that the difference between the means is due to chance) and the degree of freedom is the sum of the persons in both groups minus 2. If the result is greater than the arbitrary value in the table then the value is generally accepted as significant. The t-test formula was used to analyse the results using both weightings (R1) and (R2).

The results of both datasets (R1) and (R2) (seen in Figure 4.55) show that the test group (those using SES) out performed the control group (those using Google) when choosing and rating relevant results. However when the t-test was applied to the data set R1 (relevant results with weightings), a value of 1.726 was returned. This was lower than the designated value in the table of 1.734 which showed that the variance between the two sets was not enough to make the results significantly different. But, when the t-test was applied to the data set R2 (relevance results solely), a value of 2.307 was returned showing that the variance between the results was significantly different. This meant that users in the test study (using the SES tool) performed significantly better at *finding and selecting* relevant results than users in the control study (IE tool users), but were not significantly better at determining the actual relevance of that result.

It can be speculated that the reason for the discrepancy in the results of R1 is due to the process of rating pages. In the post-study questionnaire over 50% of users expressed their unease about having to rate a page purely based on the meta-information returned by Google. They could use the meta-information to locate what they thought was a relevant page, but could not determine the exact relevancy of the page. It is speculated that, by asking users to rate pages, a change was forced in their search behaviour and thus the results became skewed.

Another measure is to evaluate how users interacted with Google. It was discovered that on average nearly all users of the Google interface viewed results beyond the first ten results, this is shown in table 4.11. Previous experience had suggested that users of the traditional search result tools would rarely view beyond the first ten or twenty results, which is contrary to the results shown. It was theorized that by controlling the environment of users (not allowing reformulation, or viewing of web pages) users search process was unintentionally changed. Users could not reformulate their query after the first ten results, and were forced to view results further down the list.

However an important and interesting discovery was that 72% of users of the SES interface utilized the Summary view to search for relevant results from which 83% were correct. The Summary view in the SES interface was created with the aim of providing users with a quick and easy view from which similar results could be located. The results reinforce this theory, and suggest that search difference visualization is useful and enables users to more quickly drill down to interesting results.

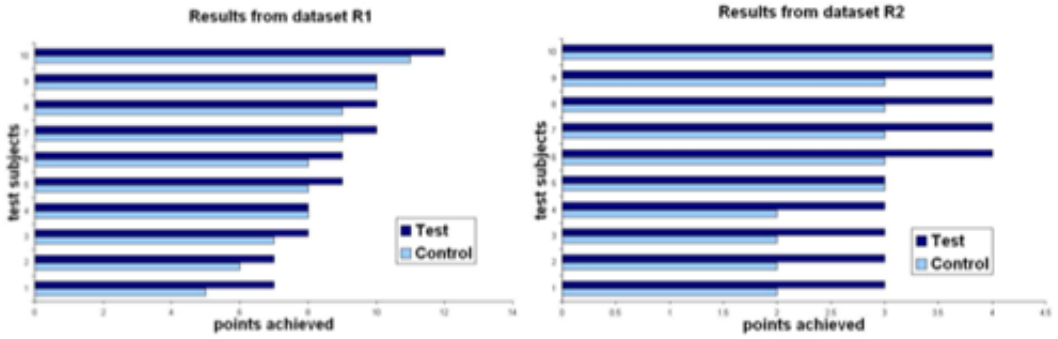

Figure 4.55: **SES experimental results.** Each bar represents a test subject (totalling 20 people), subjects from the test group (SES) are coloured in dark blue, while those from the control group (Google) are coloured in light blue. The two graphs show the results using different weightings (R1) and (R2) (left and right respectively).

This section introduced the SES interface, which in turn provided two methods of aiding users in information-seeking. These were the comparison tool, and similarity highlighting. The comparison tool displayed the website intersections of multiple searches, the theory being that any websites which existed in more than one search on a particular topic contained information on different aspects of users searches, and thus was relevant. The similarity highlighting worked in conjunction with the comparison tool, highlighting similar results (results which came from the same website, but not necessarily the same web page) between different searches. This form of highlighting augmented the appearance of interesting search results, allowing users to more easily identify and seek the information. The evaluation of the SES tool showed us the value of the comparison view, but also alerted us to the potential pitfalls of evaluating visualizations,

information that could be applid to future experiments. The next section will look at the visual-history bar, a tool designed to aid users in information recall.

4.3 The visual-history toolbar

This section discusses the implementation of a tool that aids users in recalling search history information effectively, the main concerns identified as being (1) the recalling of web pages (both in the long and short term), (2) the recognizing of web pages in the history, (3) navigating the web history, and (4) avoiding 'losing' web page trails (the aforementioned branching problem). It should be noted that the design for an evolving search history tool stores data on a session by session basis. The usefulness of tools for long term recall is not discounted (see Kaasten et al. [61]), rather, the tools usefulness for short term recall is concentrated on. An additional constraint was that the history should be compact and unobtrusive, so that searching the history would not interfere with the main web search.

4.3.1 Design

As the basis for the tool, the use of a two-dimensional layout was decided to be appropriate, much like the Hy+ browser [40], and PadPrints [47] tools. The advantage of the design was that it promoted free and and opportunistic movement among search results, allowing users to jump to (and hence recall) web pages more quickly. Referring back to the problem of losing web page trails, in using a two-dimensional space users will never overwrite their web page trails (unlike the web browsers back button). However, using a two-dimensional graphical space creates two problems (1) how will users identify objects in the two-dimensional space as the web page they wish to recall? (recognition) and (2) how can a (conceivably) large and expanding two-dimensional space be made compact and unobtrusive?

Design issue I : recognition

People remember information in different ways, and remember different types of information. For example, users can remember the layout and colours of objects in a web page, thus a pictorial representation (such as a thumbnail picture) of the website is often highly recognizable to users. Users might remember information related to the web page contents, such as meta-data (e.g. domain type, image count) or key words / unique words in the web page text. Users can identify different web pages based on different aspects of time, including time spent reading web pages and web sites. Users can remember (in vague terms) how often was spent viewing different pages and can identify which web pages were passed without much of a glance (low time spent viewing) and those that were read and browsed thoroughly (high time spent viewing).

Each of these methods differ in their suitability for use with information recall. Some information is easier to remember than others, for example, domain type is easy to recall, but the number of internal links in a web page is not. Users sometimes cannot recall the precise details related to the information they viewed, e.g. users will remember which pages contained lots of images, but not how many images were contained in each page. The different methods of recall have different levels of suitability with regards to use with two-dimensional graphical interfaces. For example, the use of a thumbnail picture to identify and recall a web page is easy and effective, however it is also unsuitable on the basis that the thumbnails picture needs to be large enough to be recognizable, a problem if the interface is constrained on size.

Finding what information people use for recall is only half the problem; methods of displaying the information in a way that helps users find what they want must also be considered. With a two-dimensional layout as a basis, the next step was to determine which perceptual variables would be best suited to aiding users in finding and recalling information. To this end, different designs were suggested in the research (seen in Figures 4.56, 4.57, 4.58 ,4.59), each utilizing a different perceptual variable, to guide users to pertinent information. Each of these variables is presented in turn.

Figure 4.56 (right diagram) shows an example of how shape can be utilized to display web page meta-data. The use of shapes to display different pieces of meta-data can be seen in other visualization works ([93], [105]). In this case, the domains of the web pages are mapped to different shapes. While this allows users to more easily identify sets of web pages belonging to the same domain, it also increases the complexity of the view, as users have to learn the associations between the shapes and domain types.

Colour can be used in two ways to help users recognize web pages, either by highlighting (and thus grouping together) related web pages (such as web pages from the same website), or to show a progression in ordinal data (such as recency). In Figure 4.57 (left diagram), highlighting is used to identify which web pages belong to the website Amazon.com. Highlighting can aid users in finding websites in which many web pages were viewed, or simply locate web pages from the same web site that were viewed at different parts of search (as the page icons would become highlighted, and easier to identify). In Figure 4.57 (right diagram), the hue of the colour changes across icons, according to the recency of visitation of each web page (darkest being the most recent).

Altering the size of the icons in a two-dimensional display is often done to draw users attention to specific points of interest. When utilizing recall information, this can be applied to identifying the time spent per web page, as well as the recency of the web pages. In Figure 4.58 (left diagram), the time spent viewing each web page is mapped to the graphical display, and the size of each icon reflects the amount of time spent viewing it. This helps users to automatically

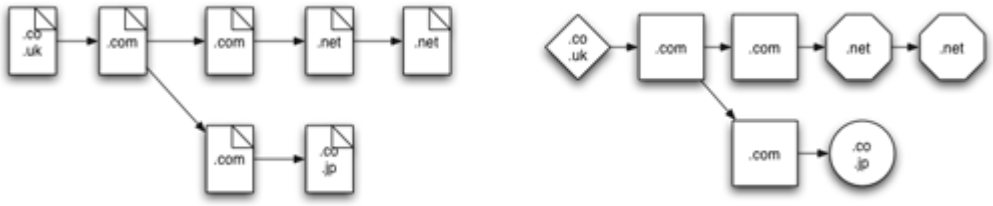

Figure 4.56: **Tree representation of a search history (left) and a search history utilizing shapes (right).** On the left is a standard tree representation of a search history that is utilized to illustrate the various perceptual variables that can be utilized with a search history. Each web page viewed is represented by a small document icon; note that in this case the domain-type meta data is represented as text on the icons. On the right it shows a view of the same search history, but utilizing different shapes to represent the domain-type metadata (much like the multi-form glyphs in [93]). While this adds an extra layer of information for the user, it also adds an extra layer of complexity, where users have to learn the mappings for the individual shapes.

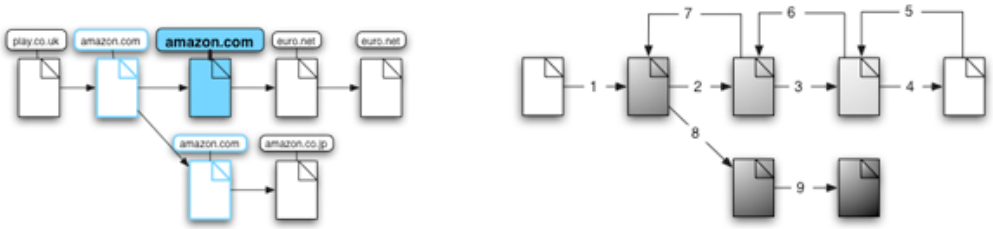

Figure 4.57: **Search history mapping hue to domain type (left) and a search history mapping hue to recency (right).** On the left is a search history using hue to show the domain type. By highlighting icons from the same website, a boundary is generated which groups the icons together, making it easier for users to recognize (and navigate back to) websites which they have viewed many web pages from. On the right is a search history utilizing hues to show how recently each web page was viewed, with the darkest icons representing the most recently viewed web pages.

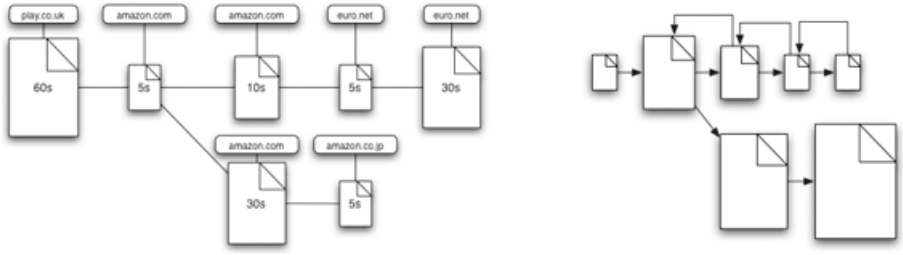

Figure 4.58: **Search history mapping size to time (left) and a search history mapping size to recency (right).** On the left is a search history which maps the time spent reading a web page, to the size of the web page's icon. This allows users to quickly pick out individual pages which they spent a long time reading. On the right is a search history which shows how recently each web page was viewed, with the largest web pages being the most recent. Note the distortion generated from the size change, which is a less than ideal trait for a compact visualization.

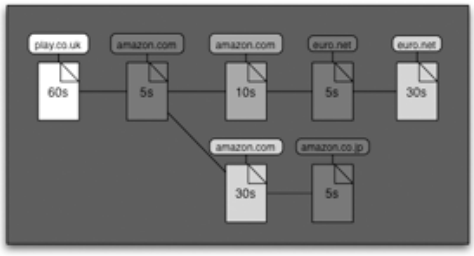

Figure 4.59: **Search history mapping time to foreground/background.** The figure shows a search history, where the time spent viewing a document has been mapped to the hue of the web page icon. The more time spent viewing the web page, the lighter the hue, this has the effect of bringing web pages which were viewed a lot, into the fore ground, while pushing web pages that were not viewed much into the background.

pick-out pages that they spent a long time browsing. In Figure 4.58 (right diagram), size is mapped to recency, where the largest pages are the most recently viewed. This helps users view their search progress, easily being able to identify which pages were viewed recently and which pages were viewed a long time ago. Notice however, that the gradual change in size causes a distortion of the display, a factor which is detrimental to a visualization aimed at minimizing the size of the display.

Creating a foreground and background effect brings important results forward, and moves less important information out of direct sight. In Figure 4.59 (right diagram), time spent viewing the web page is mapped to the icon's hue, where the lighter the hue, the more time spent viewing the web page. This has the effect of clearly identifying icons that represent web pages that users have spent a long time viewing, and moving icons of web pages that were only visited briefly to the background.

The advantage of the non-linear nature of the hyper-link structure means users can jump between web pages backwards, forwards, and even skip whole sets of pages moving from one page to another. The advantage of using the above perceptual variables is that the linear ordering of web page visits can be preserved, albeit in a different encoding variable (e.g. mapping recency to size or colour). Indeed, many of the perceptual variables mentioned can also be utilized in conjunction with a set of dynamic query tools (such as in the Grokker tool [39]), which would allow users to seek information through their search history more efficiently.

Design issue II : compaction

The visual-history tool-bar was designed with the vision of using a rectangular bar (much like a windows task bar) that could pop-up and be clicked on to opportunistically roam around the history. The traditional hierarchy tree was used for visualizing a web searcher's history on a two-dimensional space. The standard hierarchy layout was kept (as illustrated in Figure 4.60 - left diagram), and mapped to the block diagram (Figure 4.60 - right diagram).

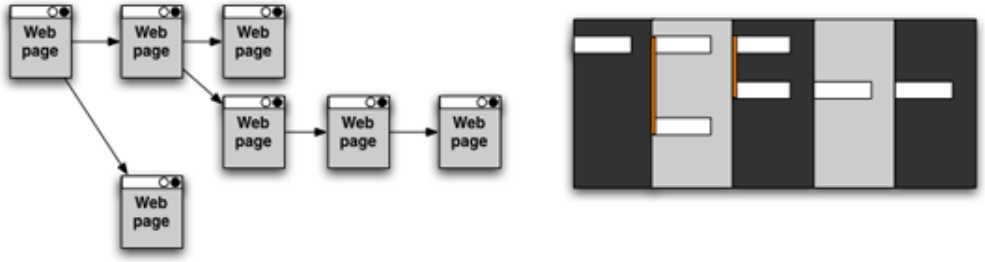

Figure 4.60: **Visual-history design.** Left, the traditional tree hierarchy of web search history (using a standard tree layout), Right, the same tree, but using the visual-history model. Note that all links that come from the same point of origin are linked by a vertical yellow line.

Figure 4.61: **Visual-history : dealing with overflow.** Left, An overflow of web pages, Right, the visual-history technique of dealing with overflow. Note that when using the visual-history technique, the light grey column extends itself, so that all nodes (representing links from a single website) are kept in the same cluster.

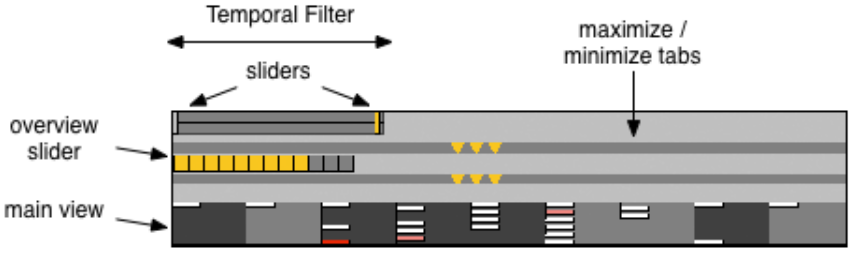

Figure 4.62: **The visual-history toolbar when minimized.** As you can there are three rectangular levels, the top is a temporal slider, the middle is the overview, and the bottom is the history itself. The top two levels (separated by grey bars with 3 yellow triangles) are tabs and can be double clicked to open up (and close) showing extra details and functions.

The only problem with utilizing a two-dimensional tree layout was the overflow of the data. If too many web pages were viewed from one page (as seen in Figure 4.61 - left diagram) then the number of bars in the column would extend past the bottom of the view. In order to deal with the overflow problem, as well as keep a compact view that would fit easily within a tool-bar, a new technique was designed which involved shifting the data along the horizontal axis of the tool (see Figure 4.61- right diagram). When overflow takes place, the data which falls beneath the tools container is placed into the next column along, and the remaining data is relocated by one column to the right. Also note that the colour of the column changes to signify that this column contains overflow information from the previous column; this is done by using the same colour for the new column, which promotes the blocks as belonging to the same set.

4.3.2 Implementation

The visual-history tool-bar combines some of the features mentioned above, most notably the two-dimensional tree hierarchy with the overflow layout, dynamic query sliders for temporal searching, input boxes for keyword searching, and an overview for when the data extends beyond the visual boundaries of the browser; this can be seen in both figure 4.62 (minimized), and figure 4.63 (maximized). The tool bar can be split into three main parts, (as shown in 4.63) at the top are (1) the dynamic query tools, (2) the middle is the overview, and (3) the bottom is the main view of the visual-history itself. The top two views begin 'minimized', but clicking upon the grey bar (with 3 yellow triangles) below each of the top two views 'maximizes' them, revealing more tool functions. Each of these views is presented and discussed.

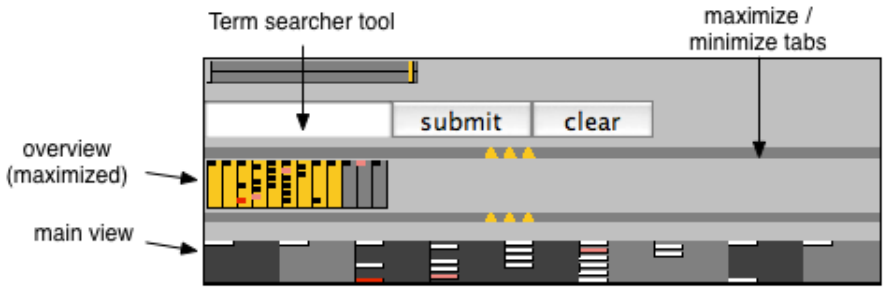

Figure 4.63: **The Visual-History toolbar when maximized.** (both tabs opened). At the top in addition to the temporal slider is a term searcher, where users can input terms to narrow their search. The middle opens up into a larger more detailed version of the overview. Note that users can start interactions with the overview by dragging the orange bar, which shows the current columns shown in the main view.

When the dynamic query tools are minimized, only the temporal slider is shown; when maximized both the temporal slider and terms searcher tool are shown. Both tools can be used to restrict the results shown in both the overview and main view, on a temporal and keyword

basis. The temporal-slider allows users to manipulate a pair of sliders which restrict the results shown on the basis of when they were viewed. Moving the left slider (grey) towards the centre begins restricting results which appeared early in the search, and moving the right slider (orange) towards the centre begins restricting results which appeared late in the search. When users manipulate the query tools, the irrelevant results are 'greyed-out', their shade becoming a closer match in hue to the background, so as to bring the results relevant to the dynamic-query closer to the foreground. The term searcher allows a similar restriction to occur, but this is instead based on keywords submitted by users in the keyword box.

When minimized, the overview is represented by a rectangle divided into small blocks, each block representing a column in the histories main view. The rectangle is normally orange, unless users generate more columns than are visible on the available screen space, in which case the non-visible columns are colored in grey. Clicking on the orange rectangle allows users to move the current view over non-visible columns. When maximized, the view becomes a miniature replica of the main view with corresponding blocks and columns.

The main view itself uses the block layout described previously. Users can click on any block to switch to that web page immediately and hovering the cursor over the block brings up a small pop-up detailing the web pages title. Blocks have 4 types of coloring, the deep red is the current result being focused, light pink are results that have been bookmarked, white results are ordinary web pages, and grey blocks have been greyed-out through the use of the dynamic query tools.

The tool was chosen to be utilized in the EvoBerry tool, because it provided a compact method of visualizing web histories. In this section, the designs for a visual-history toolbar were discussed. The tool was designed with three specific criteria in mind: (1) the tool should allow users to browse their search history opportunistically, (2) the tool should not dominate the display, being unobtrusive and able to fit on a toolbar, and (3) the tool should not have linear history system (so as to avoid the branching history problem). The tool design addressed the issue of *information recall*, allowing users to quickly and efficiently retrieve their history information.

4.4 Information management tools

Evolving searches generate a large amount of information, that in turn must be managed effectively to allow users to keep track of important threads of data, as well as allowing them to instantly recognize and recall previously viewed data. The work space must also be managed effectively to allow users to create an effective work environment, where important tools and information are always close to hand. Information management tools can vary greatly in scale

and appearance. They can be as simple as a set of coloured highlighting or as complex as a desktop environment. More often than not, an information management tool is not the focus of the interface, and only fulfills a support role. As a result, an interface consists of several different management techniques working side-by-side to support users search process. This section looks at the different information managements techniques that will be incorporated into the evolving search interface. The objective is to provide (1) a workspace that can be organized by users, (2) techniques for automatically organizing users search data, and (3) tools that allow users to manually organize their search data.

4.4.1 Design

There are two areas where information management is most important: (1) in the workspace and (2) in the data. The management of data can be broken up into tools that manage data automatically, and tools that manage data manually. These are each discussed in turn.

Workspace management

In order for effective workspace management to take place, a tool must have a flexible design. The previous mentioned interface designs (visual-bracketing tool, coordinated-bracketing tool, SES tool) have all utilized a fixed layout, and as a result are not flexible in the positioning of their functions and data. Ideally, a desktop workspace would allow a more flexible layout, where users could arrange the layout of their tools and data within the interface. Switching to a desktop workspace allows different interaction and display techniques to be utilized. For example, the fixed workspace of the SES tool only allows 3 searches to be compared at a time, whereas utilizing a desktop workspace would allow any number of searches to be generated, and compared at the same time.

Data management

In the design chapter, the benefits of providing both automatic and manual data management systems were mentioned. In an evolving search, data management is important, because it allows users to keep track of the different search threads currently being engaged in. Simple procedures could be put in place to automatically tag and identify information from different threads, for example, a coordinated colour system where objects belonging to specific threads are assigned specific colours. This would allow users to quickly locate and identify related information. Ideally, users would be allowed to change the colour mappings of the different threads e.g. using similar colours to group together similar searches and assigning different colours to disparate searches.

In addition to the managing of large sets of data, users should also be given the ability to organize individual pieces of information (individual notes, web pages, and search results), much like a bookmarking system, but more broad in nature and use. This spawned the idea of a 'drop-box', which users could (literally) drag information from any of their sources (be it search result lists, highlighted text or whole web pages) and 'drop' into the box. Once inside the drop-box, the information can either be left in a 'receiving area', an unorganized area where all new information is placed, or be managed (under the direction of the user) and placed into a specific category or folder. Ideally, the drop-box would search and filter results in the drop-box according to data types, meta-information and keyword tags.

Another aspect of automatic data management is placing information where users can find it easily and retrieve it as necessary. This means that, not only must related data be visible, but it must also be located in close proximity to each other. Combined with the aforementioned desktop workspace, this gave rise to the idea of an integrated data window, where all the information related to a particular search would be contained. This meant that each window would not only display search result information related to a particular query, but would also contain information about each search result and web page opened from that search result list. This would allow users to keep track of, and recall information more quickly, and it works well with the previously discussed colour coordination. This integrated data window was named, the 'results frame', because its primary use was to contain the search results returned from the query. The word 'frame' (in 'results frame') was derived from the Java containers used on desktop interfaces, which are called 'Internal frames'.

This section investigated the use of different techniques to aid users in managing their information. Specifically, four techniques were identified that needed to be incorporated into the design: (1) a desktop workspace, (2) colour coordination of search threads, (3) a drop-box for manually storing information, and (4) an integrated data window (results frame). The techniques described addressed the issue of *information management*, allowing users to manage their workspace and data for a more efficient identification and recall of information.

4.5 Overview

This chapter looked at the different designs and techniques that would be integrated into the evolving search interface. In particular it focused on techniques and tools that addressed the issues of (1) information visualization, (2) opportunistic searching, (3) information recall, (4) information-seeking, and (5) information management.

The visual-bracketing tool utilized the visual-bracketing technique that applied a focus-and-context view to search result data. It allowed the visualization of large amounts of data as well giving users the ability to browse search results opportunistically. This also introduced the concept of using multiple views to produce different views of the data.

The comparison tool allowed the comparison of search results which existed in multiple searches. The tool provided a method of seeking interesting information through the comparison and augmentation of similar search results. The visual-history tool introduced a novel method of displaying a search history. The design allowed users to browse opportunistically, and avoided the problems of previous history systems (such as branching). The tool was designed to be compact and unobtrusive.

In addition to these tools, several information management techniques were discussed. The use of a desktop workspace was considered because of increased flexibility over a fixed workspace, which is important in the visualization of multiple searches. Techniques for both automatic (colour coordination) and manual (drop-box) management of information were discussed. Colour coordination could be used to group together information from the different search threads. A drop-box would allow users to manually store and categorize information from different sources for future recall. An integrated data window was suggested (as part of the desktop interface) that would group together information from different search threads, allowing users to more easily locate information to be recalled.

The three tools, as well as the information management techniques discussed, all formed the basis for the final design of *EvoBerry*, the evolving search and berry-picking tool. During the design process, the evolving search interface went through a prototype stage and a final implementation phase. Although the prototype was generally thought to be inferior to the final version, it contained some tools made unavailable in the final version, due to time constraints. In the next chapter both the prototype, and final implementation of the EvoBerry tool will be discussed, along with different technical challenges of implementing such a tool.

Chapter 5

EvoBerry

There were two main differences between the EvoBerry interface and previous interfaces (such as the *coordinated bracketing tool* and the *SES tool*). First EvoBerry made use of a desktop-style workspace (similar to the desktop interfaces used in both the Windows and OS-X operating systems). This was part of a move towards a more flexible layout that would help users manage the data and tools within the work space. This increase in *secondary notation* (user's control over their displays and the arrangement of their workspace) is supported by authors (Hendry et al. [46], Petre et al. [12], and Kirsch [62]). The second main difference was the data structure employed. With the integration of several different tools, and the inclusion of history and recall tools, it was necessary to develop a universal data structure that would be employed by all the tools. A diagram of the data structure can be seen in Figure 5.64. The data is organized into a hierarchical tree which encompasses a search session. Starting at the top, the search process can be broken up into several *search threads* (generated every time users input a query). Within the search thread, there are several *search sessions* (generated every time users open a search result). Within the search sessions are several web pages.

Search threads are the data objects that sit at the highest level. A search is comprised of several search threads, such that every time a search is input into the search engine a new thread is generated. This structure was borne directly from the evolving search theory, where with every search performed, a user's information need is changed. It was theorized that each search would take a slightly different direction and as a result would deal with slightly different information than the previous search. Thus all information generated from a single search is considered to be part of the same group. Search threads consist of any number of search sessions and web pages.

Every time users open a search result, a search session is created. Theoretically, all the web pages that users view from a single source (the starting web page) should be related, e.g. each

web page was discovered by following a link from a previous web page that (at some point) originated from a search result. Thus all web pages opened from a single search result can be grouped together as being similar. A search session can consist of any number of web pages. The lowest level of data object is the web page.

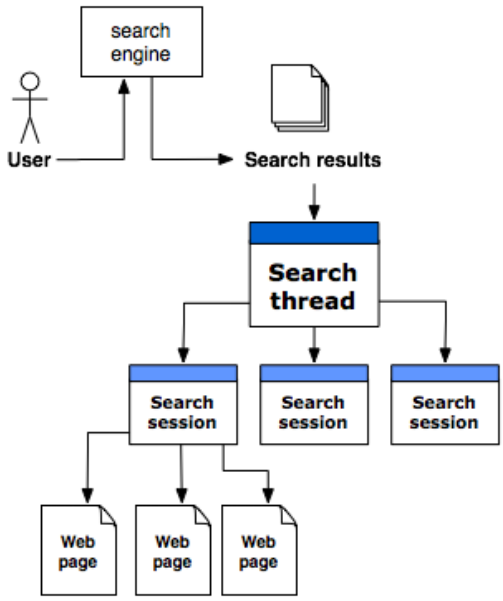

Figure 5.64: **Data Structure:** This is a data structure developed specifically for the evolving search tool. When a search is generated, a set of results are returned. These results form the basis for the *search thread*. A search thread is the highest level object in the structure, it contains a set of search results, and is identified by the query used to gather the results. Whenever users opens a search result from the search threads result list, a *search session* is created. A search session contains all the web pages viewed, which originate from an initial search result.

The only criticism of this data structure is the fact that sometimes searches performed by users will all follow the same topic, and in their mind are all part of the same group. Ideally, in this situation users group them together under the same thread. However, this structure forms the basis for the automatic organization of data, and as such, by default, each search generated becomes a thread. This lowers the cognitive overhead that would take place if users had to manually categorize each search as it was generated. Ideally, this automatic structuring system would act in the background, while still allowing users to customize the information manually, and reorganize and re-categorize threads.

A prototype of the EvoBerry interface was developed as a precursor to the final version. Both utilized the same layout (desktop workspace) and data structure, but each differed in certain aspects, such as data display and tools made available. This chapter discusses the EvoBerry interface, first presenting *the prototype version*, looking at the different tools and techniques used in the design. Next a *technical discussion* is presented, of the programming required to build the underlying tools, as well as some of the technical challenges of building such a tool.

The last section presents *the final version* of the EvoBerry interface, discussing the changes that took place between the prototype and the final version.

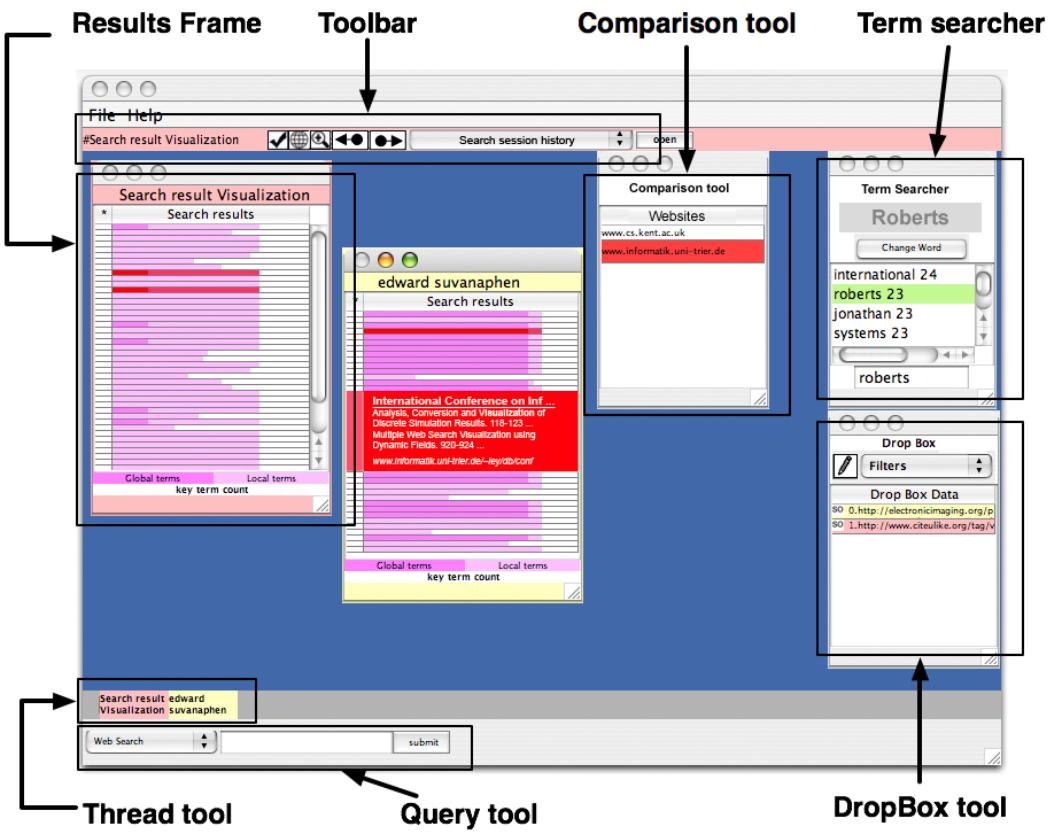

Figure 5.65: **EvoBerry prototype.** The prototype can be divided into three separate parts, the tool bar (top), desktop (centre), and search bar (bottom). The toolbar varies in appearance based on which frame is being clicked on. In general it contains buttons that allow interaction with the data within the frames, e.g. buttons to open up web pages, and bookmark results. The bottom contains the search bar, which allows users to manage existing searches, as well as perform new searches. The centre is dominated by the desktop which contains all the data and support tools.

5.1 Prototype version

Figure 5.65 shows a view of the prototype interface which can be split into three primary parts, (1) the search bar (at the bottom of the interface), (2) the desktop (centre of the interface), and (3) the tool bar (at the top of the interface). The search bar contains two tools, the query tool and the thread tool. The query tool is where users input their searches. When a search is submitted, a set of search results is returned, and then a coloured block (representing that search) is added to the thread tool.

When a search is performed a set of search results is returned to the desktop. The results are placed into a results frame on the desktop for users to inspect. Three other frames exist on

the desktop: these are *tool frames* (the drop box, comparison tool, and the term searcher).

The toolbar contains a set of buttons that can be used to perform different actions upon the results, e.g. open web pages, bookmark results, zoom in/out. Note that the buttons available on the toolbar differs based on the frame that has been currently selected, since different frames have different functions.

One other view, not shown here is the web browser view. When a search result is opened, the corresponding web page is sent to a Java web browser window for displaying (based on the JDIC web browser [23]). This section will look at the prototype tool in detail. The prototype was written using the Java (1.4) programming language [118], and utilized the Java interface to the Google Web API [3], in order to get the search result data. The underlying technology behind the prototype tool is more thoroughly discussed in the technical discussion section. Each of the parts of the interface mentioned above, as well as the Java web browser used for viewing web pages, will be discussed in detail.

5.1.1 View I : search bar

The search bar sits at the bottom of the interface (see Figure 5.65 - bottom), and contains two tools, the query tool (bottom of the search bar) and the thread tool (top of the search bar). Users begin the search by inputting search terms into the query tool, and pressing submit. A search request is then sent to the search engine, which in turn returns a set of results which are visualized on the desktop as a results frame. On the search bar is a small drop-down box, which allows users to change the type of the search. There are four principal search types, the web search, the title search, url search and phrase search.

The web search is the standard search; it simply sends the search terms straight to the Google web service, and returns a set of search results. The title search restricts the search to only words in the title of the web pages. This is useful if users are attempting to recall a web page, but can only remember the title. If the search terms [africa safari] were being used for the search, then switching to a title search would be equivalent to adding the Google modifier [allintitle] (e.g. [allintitle: africa safari]) to the query.

Selecting a URL search, restricts the search to only words found in the URL of the web page. This is useful if users are attempting to recall a web page, but can only remember certain parts of the URL. For the search [safari], switching to a url search would be equivalent to adding the Google modifier [allinurl] (e.g. [allinurl: safari]) to the query. Selecting a phrase search, restricts the search to a particular phrase, which has to be matched exactly. This is useful if users want to match a quote, or want to make sure that the stop words are not removed from the search. For the search [the big five], switching to a phrase search would be equivalent to adding a pair of quotation marks [“ ... ”] (e.g. [“the big five”] to the query.

The thread tool also sits on the search bar. Each time a search is made, a thread is generated (see the data structure in Figure 5.64) and a thread icon (a coloured block) is added to the thread tool. Each thread is assigned a unique color when it is generated, and this colour is used for every frame and information container which is borne from this search. This helps associate and group together data from the same thread. Clicking on the thread icon will minimize/maximize all the items currently open from the same thread, helping to organize the data and reduce visual clutter.

5.1.2 View II : desktop

The desktop is where all the main interactions take place, and forms the bulk of the tools view. Like other desktop interfaces, there exist windows upon the desktop, which in turn can be minimized, maximized, resized and closed depending on the user's wishes. The greatest strength of the desktop interface is the ability to reposition the windows, containing both data and tools with minimum fuss. Note that throughout this work, the term *frame* is used instead of window, when referring to a container on the desktop. This reflects Java programming conventions [118], where the containers are called frames (e.g. JFrame, JInternalFrame).

As stated previously, whenever users perform a search, a *results frame* is added to the desktop, containing the results of the search. The tool starts with three *tool frames* already on the desktop by default, the *comparison tool frame*, the *drop-box tool frame* and the *term searcher tool frame*. Additionally, interactions with the data in the results frame can generate *link tool frames*, which contain web pages linking to and from a web page. When users wish to add a textual note to the drop-box, a *notepad frame* is created on the desktop. Each of the frames will be looked at in turn.

The results frame contains the search results returned from the search engine. The query entered into the search engine is displayed as a title at the top of the results frame. The search results are ordered as a list, according to relevance. Search results have two states: (1) the graphical state (see Figure 5.66 - left), displaying a pair of overlapped bars, which happens when the result is not currently in focus (has not been clicked on) and (2) the textual state (see Figure 5.66 - right), displaying textual information about the selected search result (which is now in focus). A description of the display and interaction techniques is given later on.

The comparison tool frame contains the *comparison tool*, which is used to find similar search results which exist in multiple searches. The comparison tool matches search results that were generated by different searches, but which come from the same website, and share the same base URL. These 'similar' results are ordered in a list, represented by the website's URL, with each object ordered according to the number of intersections (threads that it exists in) that it contains.

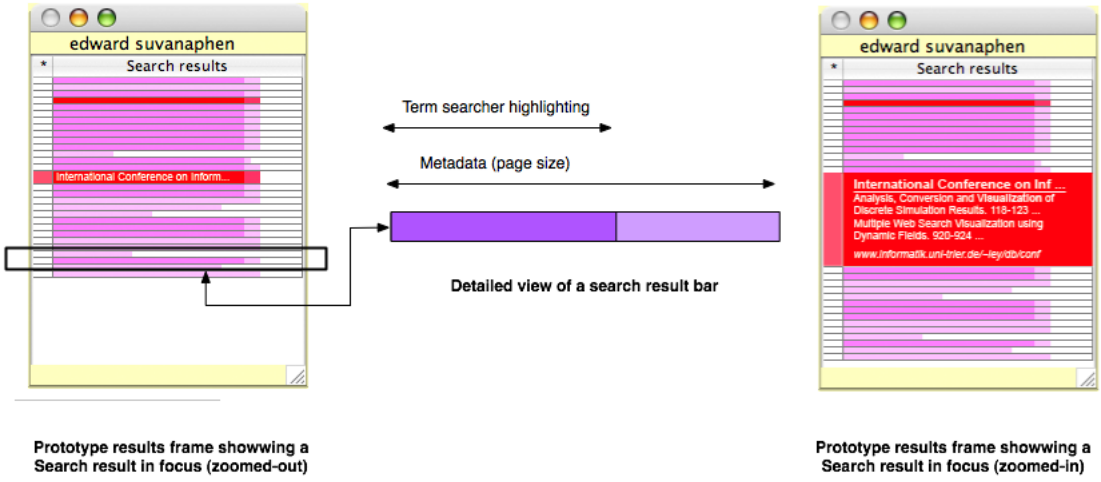

Figure 5.66: **Prototype results frame:** This shows two views of the prototype results frame, on the left is the normal view, and on the right is the zoomed-in view. Each bar has a length corresponding to a specified piece of metadata (in this case page size). The inner-bar is part of interactions with the term searcher tool. When a keyword is specified in the term searcher, then the inner bar is generated in each search result that contains that keyword. Each inner-bar has a length equal to the number of occurrences of the specified keyword in the web page.

The term searcher tool frame contains the *term searcher tool* that is used to find search results that contain specific keywords. The tool contains two functions through which to filter search results: (1) an automatically generated list of suggested keywords (based on an analysis of the returned search results data), and (2) a textbox, through which users can specify a set of keywords to filter with. The list of suggested keywords is filtered for stop-words and each entry in the list of suggested keywords, is represented by a word, and a number specifying the number of search results that the word appears in. The results are ranked according to the number of search results they appear in. The term currently being used to search with, is written at the top of the frame.

The drop-box tool aids users in managing the search result data, allowing them to store, search results, web pages, addresses and notes, to be recalled later. Objects added to the drop-box are represented in a list format, with one entry for each object. Each entry is represented by a specific reference number, a piece of text from the piece of information, and a symbol stating what type of information it is. The entry's colour is dependent on the thread it originated from. The tool contains a function for adding text notes made by the users, as well as a function for filtering the displayed objects according to type (e.g. URLS, search results, web pages and notes). These functions are represented by a note button and drop-down box respectively.

The link tool frame can contain one of two types of data, either web pages which *link to* a particular web page, or web pages that are *linked from* a particular web page. These are referred to as *LinkTo* and *LinkFrom* frame respectively. Link tool frames are generated when users press

either the LinkTo or LinkFrom buttons in the toolbar, which in turn create a frame containing a list of all the web pages that link to or from the web page of the search result, currently in focus. The link tool frames share the same display techniques as a normal results frame. The functions of both link tool frames are discussed in more detail in the toolbar section.

When users wish to write down a piece of text as a note, the 'create note' button on the drop-box (symbolized by a pencil) can be used to generate a notepad frame, a plain frame upon the desktop which has a space to write text, and a submit button. Once the submit button has been pressed, the frame is closed and the data is added to the drop-box. The different frames each have different functions and different forms of interaction, and furthermore interact with each other in different ways. The next section looks at these interactions in more detail, as well as the display techniques used for each.

Frames

The display of the **results frame** varies depending on user's interactions with it. As stated previously, a search result has two states, selected and not selected. When selected, the search results textual information is displayed (the title in this case), and when not, the search result is displayed as a bar whose height is determined by a specified piece of metadata. The data used for the bar height can be mapped to many different variables, such as page size, image count and link count. These two states form the basis of the visual-bracketing technique that is applied to the search result list. However these are only the basic states, there are three other special states that appear, when users interact with the zoom tool, term searcher tool, or the comparison tool.

Interactions with the zoom tool in the toolbar can change the display of the data currently in focus. By zooming-in the data in the focus increases (displaying the title, the search results snippet of text, URL and page size). When the term searcher tool is used to find a specific term of interest, then a set of secondary inner bars appear, sitting in the foreground of the existing search result bars (see Figure 5.66). The height of the inner bar is based on the term count of term currently specified in the term searcher tool. The results frame implements a system called *similarity highlighting*, which is used to link search results that originate from the same website. When a search result is selected, the URL is checked against those stored in the comparison tool. If a match is found, then the selected result, as well as all other results that share the same website, are highlighted in red.

When a results frame is created, it is assigned its own unique colour, which all other frames and information related to that results frame use. This colour coordination helps group together pieces of data that originated from the same thread. Each results frame is assigned its own toolbar, which appears when the results frame has been clicked upon. From the toolbar, more

interactions can be made with the results frame data. These are interactions discussed in the toolbar section. The results frame also interacts with the drop-box. Information in the results frame can be placed in the drop box, in one of two ways (1) selecting and dragging a search-result into the drop-box, or (2) right-clicking on a search result will activate a pop-up menu, where users can choose what information to send to the drop-box. Information placed in the drop box can be recalled by double clicking on its entry inside of the drop box, which will in turn open the textual information of the entry in a notepad frame on the desktop (which can then be edited).

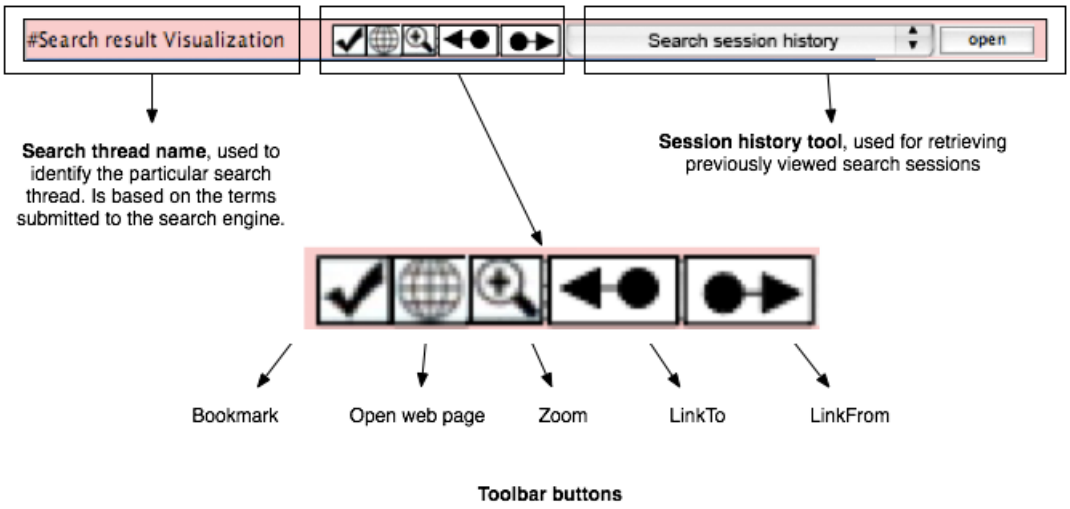

Figure 5.67: **Prototype toolbar:** Each results frame has an associated toolbar, which becomes active when that results frame is clicked upon. The toolbar consists of a the search thread name, buttons for interacting with the search result data and a drop-down box containing the different search sessions that users generate.

5.1.3 View III : tool bar

The toolbar is located at the top of the screen, and is used for interactions with the data in the results frame. Each results frame has its own toolbar, which is colored according to the thread (and results frame) that it belongs to. There are five buttons and a drop-down box on the toolbar; these are used for the *bookmark function*, *web browser function*, *zoom function*, *LinkFrom* function, and *LinkTo* function (see Figure 5.67) . The drop-down box is a *search session history*, and contains a list of all the user's search sessions in this thread. Each function is discussed in turn.

The bookmark function highlights the currently selected web page in green. This makes the result stand out from the surrounding search results, making it easier to retrieve later on in the search. Clicking on the bookmark button multiple times, while selecting a result, switches the

highlighting on and off. The web browser function opens the currently selected search result's web page, inside a Java web browser window. When this happens, an entry is stored in the search session history, so that it can be recalled later. Both the search session history and the web browser are discussed later in this chapter.

Use of the zoom in/out button, as described previously, changes the amount of information available in the focus of a results frame. Clicking on the button multiple times will switch between the zoomed-in and zoomed-out states. When users click on the LinkTo function, a link tool frame is added to the desktop. The frame shares the same display and interaction techniques as a normal results frame. The results gathered in a LinkTo frame are all the web pages that link to the URL of the currently selected search result. These are obtained through use of the Google function, [link:], which creates a search that specifically targets web pages that link to the specified URL e.g. searching for web pages that link to [www.safari.com], would send the search [link:www.safari.com] to the Google web service.

When users click on the LinkFrom function, a link tool frame is added to the desktop. The frame share the same display and interaction techniques as a normal results frame. The results gathered in a LinkFrom frame are all the hyperlinks gathered from the web page, that lead to external sources (other web pages, outside of the current website). The search session history stores all search sessions as entries in a drop-down box. Whenever a search result is opened in the web browser, an entry is stored in the search session history. When users click on the search session history, the drop-down box opens and a list of links are displayed, one for each search session. Each entry in the list consists of a unique number assigned to the search session and the title from the first web page in the search session. Clicking on the 'open' button will open the currently selected search session in a new browser window. The functions of the results frame can also be accessed by right clicking on a search result, opening a popup which allows users to use the bookmark, web browser and zoom functions.

5.1.4 Java web browser

Whenever a search results web page is opened, a new web browser window is created external to the program. The web browser was designed using the JDIC toolkit [23], which allows the creation of Java based web browsers, and uses Internet Explorer to render its web pages inside a JFrame. A view of the web browser's toolbar can be seen in Figure 5.68. There are two rows of functions in the web browser. The top row of functions are the standard web browser interaction tools, consisting of a set of buttons (stop, refresh, forwards, backwards etc.) and a url address bar. All of these functions, operate exactly as in a normal web browser. The second row contains a design of the *visual-history toolbar* (mentioned in chapter 4 - section 4.3).

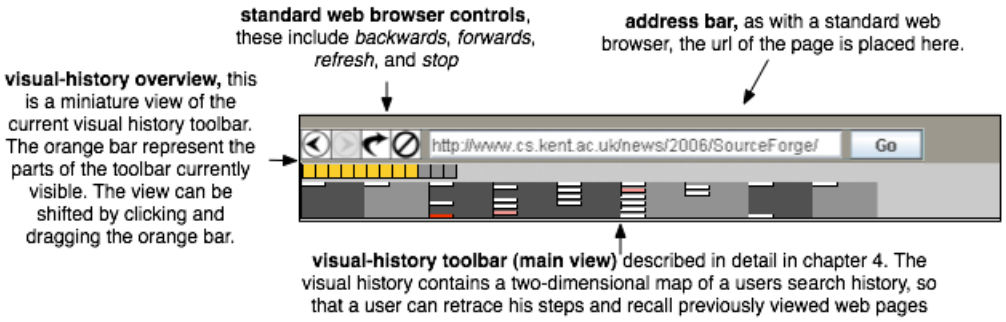

Figure 5.68: **Prototype web browser toolbar:** This shows a view of the prototype's browser toolbar. In addition to all the standard web browser functions, there is also a visual-history toolbar and overview attached (see chapter 4 for more details on the visual-history toolbar). The toolbars main view and overview can extend as far as the web browser window it is linked to. The overview can be used to scroll the main view.

The version of the visual-history toolbar that was added to the browser window was a cut-down version of the original design. Instead of including all the tools and views specified in section 4.3.2, only the main view and the minimized overview were included. As with the previously described visual-history toolbar design, every time a web page is viewed, a block is added to the toolbar, which in turn forms a tree-like structure, which users can navigate opportunistically. The page/block currently being viewed is highlighted in red. Clicking on a block opens up the web page that it represents in the main web browser view, and moves the focus to the new block. If the toolbar extends beyond the width of the Internet Explorer window, then the overview can be used to scroll the visual-history bar along, to view the hidden columns. Moving the cursor over a web page block in the toolbar will generate a pop-up with that web pages title, so users can easily identify the web page blocks. When a session is recalled from the search session tool, the complete web page history for that particular session is also returned and displayed in the visual-history toolbar, allowing users to revisit web pages easily.

5.1.5 Overview

The prototype interface introduced a workspace containing different tools that could be used for aiding users in performing evolving searches. The aim was to address the evolving search interface issues, described in chapter one. The way each tool helps users deal with the different interface issues is presented.

The search result information is visualized in different ways. Replacing the original text of search results with two-dimensional bars reduced the screen space taken up, and provided a format where metadata can easily be compared. The visual-bracketing technique was applied to the results to give two semantic levels (the focus, in text, and the context, as bars) which allowed users to view more data while still being able to read the specifics on each search result.

Similarly, the visual-history toolbar provided a graphical representation of the search history in a compact toolbar. These aspects of the tool all contribute to visualizing the large amounts of data generated.

The addition of the visual-bracketing, combined with the bar representation, has allowed a more compact display, which in turn allows users to browse the data opportunistically by jumping from one part of the results to another with a single click. Similarly, the graphical representation of the visual-history toolbar allowed users to move opportunistically over the search history.

The prototype organizes the data in a way that allows quick and easy recall, and also implements tools to aid the users in recalling specific information. Each web browser session that was created was stored within the thread of its specific session history tool, and could easily be recalled. Recalling a session also loaded up that session's web page history (using the visual-history toolbar), so that users can recall specific web pages simply by clicking on them. Manual history tools also exist that allow users to mark specific search results for recall later. This can be done in the short-term (using bookmark highlighting) or in the long-term (placing a result in the drop-box).

Tools were developed to aid users in seeking interesting results. The comparison tool and similarity highlighting allowed users to view the intersections of multiple searches. The term searcher permits users to filter search results based on specific key words. Filter tools provided in the drop-box, allow users to find entries based on type. Functions that allowed users to draw upon information from different sources also exist. For example, the LinkTo and LinkFrom frames both allow users to draw upon information leading to and from a specific web page, as well as generate information faster than if users had hunted the links down normally. The query tool can be used to specify different types of queries (e.g. title, phrase, URL).

The desktop workspace chosen allowed users better workspace management as well as more control over the positioning of data and tools. Thread coloring automatically links data together, and allows users easier recall. Data can be stored and manually managed through the drop-box. Better workspace management, as well as reduced visual clutter, is achieved through the thread tool which allows users to quickly minimize all frames and windows associated with a specific thread. The EvoBerry prototype was the initial attempt in this research at solving the evolving search issues laid out in chapter one. Through the use of an informal evaluation, problems and issues were discovered with the interface and technology.

Limitations

The evaluation lead to the discovery that, even after explaining the functions of the tools, users were still confused as to the meaning of the bars in the results frame and were displeased with

this new display format. While users enjoyed using the term searcher tool, the meaning of the size of the bars was initially lost to them. It was felt that replacing them with textual titles, and a form of highlighting (for the term searcher tool) would be more appropriate. Research has shown that graphical representations can be useful in search result visualization ([45], [70]), however it must complement the existing visualization structures.

The evaluation also identified a problem with the use of an external web browser window. It was found that having to continually switch between the main view and the various browser windows incurred a high cognitive load, much to the dissatisfaction of users. Ideally the web page would open as a frame on the desktop, however the web browser toolkit utilized did not allow this due to a problem with mixing Java light-weight and heavy-weight components (see the technical discussion in section 5.2.2 for more details). This problem is addressed in detail in the final version of the tool.

It was also discovered that the implementation for the LinkTo function was cumbersome to use because (1) it required performing a new web search to acquire the data (which in turn took some time to process), and (2) many times a LinkTo search would return no results, since the information may not have been stored on Google. Users also commented, that by placing this information in a separate frame, it created a void between the two pieces of essentially linked information.

The term searcher tool was highlighted as another problem area in two respects. (1) The search tool orders the list of suggested keywords by term count, and (2) the inner bar's height was based on the term count. It was later discovered that the use of search term count as a measure of relevance is an erratic and unpredictable method at best. For some pages the number of times a term appears in a page indicates its relevance in the field, e.g. a website advertising holidays will have the word holiday appear many times; However, very often 'stop words' (see section 3.1.1) will appear more often than non-stop words. Even after filtering these stop words, common non-stop words will appear in great quantities e.g. in the search [web visualization], both web and visualization will appear many times. The problem is that, quite often, the more interesting keywords will be unique, and very often only appear once or twice in the web page. For example, the name of an author in a given area of research is often important for seeking related research papers, however the name will very often only appear once or twice on the author's home page. This showed the term searcher tool as inefficient in aiding users to find relevant web pages.

These observations formed the basis for the redesigning of the EvoBerry interface. During the implementation of the EvoBerry prototype, technical problems with the interface were noted. The solutions to many of these problems directly contributed to the changes made in between the prototype and the final version of the EvoBerry interface. The next section looks in detail

at the underlying technology behind both versions of the EvoBerry interface, as well as the technological challenges that needed to be overcome to produce the final version of the interface.

5.2 Technical discussion

This section presents the technical aspects of the EvoBerry interface in two parts, (1) the *technologies used* in designing the EvoBerry interface, and (2) the *technical challenges* that needed to be overcome in order to produce a fully functional evolving search tool. It should be noted that both the prototype and the final version of the EvoBerry interface shared many common technologies. This section will use the name *EvoBerry interface* to refer to both of the tools, and will specify explicitly when referring to one prototype or the other.

5.2.1 Technologies used

The EvoBerry interface was written in the Java programming language [118], with Java 1.2 for the initial prototype, and version 1.4 to write the final version. Java was used to create the GUI interface, the data structure for storing the data, and provided various 'listeners' that allowed users to interact with the interface. The only two services that were not provided as part of the standard Java development kit were the abilities to gather search results from a search engine, and render web pages within Java containers. This section discusses the different technologies involved in the design of the EvoBerry interface, looking at each of the following areas: (1) the interface display, (2) underlying data structure, (3) interaction techniques, (4) data gathering and (5) web page rendering.

The interface display was primarily built using the Java *Swing Toolkit*, although a small amount of the *AWT Graphics* package was used to design the final version's session bar and the visual-history toolbar. The Swing toolkit was invaluable, as it gave access to interactive tools such as the JDesktopPane which allowed the creation of a desktop-type interface. The Java collection classes (HashMap and ArrayList) both provided useful pre-built models for data storage, and were used to develop the data structure required. Data was organized according to the structure specified in the introduction to this chapter (according to search threads and sessions). Data was stored in two structures, the *DataPool*, and the *ConfigFile*. The DataPool was used to contain permanent, unmodifiable data, such as the search results returned from the web service. The ConfigFile was used to contain the temporary variables that change when users interact with the interface, such as the location of the search result currently in focus, and the number of search threads currently being displayed.

A large number of *EventListener* objects were used to allow users to interact with the tool. This provides a wide range of functionality such as clicking (left and right), scrolling and

dragging of objects. Different listeners, combined with renderer objects, were required to create the visual-bracketing technique. For data gathering, at the time of the design of the EvoBerry prototype, only Google [3] provided a web API with a Java interface. This was far more ideal than the alternatives which was to write a program that performed the search and harvested the information from the HTML pages of search results returned. This was far less efficient and would take longer to obtain results. Near the end of the development of the final version of the EvoBerry interface, Yahoo! [127] introduced a Java compatible web API. It was decided to use this service because the Google web API had not been updated for a long period of time, and was frequently returning errors.

The Java SDK provides methods to retrieve and display simple HTML pages using the JEditorPane. However rendering more complicated web pages utilizing non-standard web technologies such as Javascript, applets and Macromedia Flash content was not currently supported, neither was the display of non-HTML file types such as PDF documents. This prompted a search for a Java library that allowed a fully functional rendering of a web page. The JDIC web browser [23] was discovered, which utilized existing web browser renderers (Mozilla web browser[32] and Microsoft Internet Explorer [90]) to render web pages within Java containers. However, there are still problems and challenges to effectively integrate this technology into the EvoBerry design (detailed in the technical challenges - section 5.2.2).

The design of the display, and techniques utilized, were greatly dependent on the data available for each search result. Both the Google and Yahoo! APIs provide a set of standard search result information as part of the data returned for each query. Typically they return the following information: (1) the title of the web page, (2) the URL address, (3) the size of the page (in kilobytes), and (4) a snippet of text, showing the query terms in context. Additional information can be obtained through the API, but this would require users to perform specific additional searches, e.g. obtaining a list of all web pages that linked to a specific URL requires a special secondary search to be performed. In addition to this, different types of search can be performed to filter the data returned, typically these include (1) a title search (limiting the search to words in web page titles), (2) a url search (limiting the search to words in url addresses), and (3) phrase searches (the search results must contain a specific phrase). This is in addition to all the boolean operators that can be applied to a standard search engine.

Many of the tools of both the prototype and final version of the EvoBerry interface require information that is not available through the APIs. In the prototype, information on the different keywords present in each web page was required to use the term searcher tool, and the LinkFrom function requires knowledge of all the links in a web page. The final version of the EvoBerry interface also utilizes knowledge of a web page's hyperlinks for use in its overview, as well as information on the amount of text and images present in each web page. As a result, it was

necessary to extend the data gathering process beyond the standard processes presented through the API; as each search result was returned, its URL was extracted, the page looked up and retrieved. Each page was parsed to gather information, including the structure of the web page, frequency analysis of the page, as well as further keyword and HTML tag information. While this extended the range of information available to use with the interface, it also generated its own set of problems (discussed in the technical challenges section).

5.2.2 Technical challenges

Several problems had to be overcome in order to implement a functional version of the EvoBerry interface. These problems can generally be divided into two areas: (1) display problems, and (2) speed problems.

Display

One of the essential components of any evolving search interface is the ability to allow users to browse web pages. The initial tools developed (such as the coordinated bracketing and SES tool.) forwent the need for an integrated web browser, instead using a Java RunTime function to activate the operating system's native browser as an external window and open web pages within that program. However this generated two problems: (1) the web pages were being opened in separate windows external to the program, and (2) the functions and data of the web browser could not be integrated into the search tool. Integrating the web browser view into the main search tool was considered a priority because of the increase in cognitive load that developed from having to switch between multiple browser windows and the search tool. Additionally, the evolving search tool aimed to help users in web page browsing and history recall, which meant access to information specific to the browser (e.g. which web pages were viewed, and in which order) was needed.

As mentioned previously, utilizing the JEditorPane to view web pages was not ideal given the inability to view web pages which utilized non-standard web technologies; an integrated Java web browser was needed. At the time of design, the JDIC Java web browser [23] was one of the few Java libraries available that allowed the creation of Java based web browser containers, which could be easily utilized as part of a Java interface. However, it was quickly discovered that a problem existed in mixing the JDIC web browser object (a Java *heavy-weight* component) and the JInternalFrame's used on the JDesktopPane (both Java *light-weight* components). The problem stems from the mixing of light-weight and heavy-weight components [33]. A heavy-weight component is one that is associated with its own native screen resource, while a light-weight component is one that "borrows" the screen resource of an ancestor, meaning it has no native resource of its own. It is generally recommended not to mix these two types of components

since there are significant differences between light-weight and heavy-weight components that become painfully obvious when the two are mixed. Below are a list of the most common differences:

- **transparency:** light-weight components can have transparent pixels, whereas heavy-weight components are always opaque.
- **shape:** light-weight components can appear to be non-rectangular (because of the transparency) whereas heavy-weight components are always rectangular.
- **mouse events:** mouse events on a light-weight component fall through to its parent; mouse events on a heavy-weight component do not fall through to its parent.
- **overlap:** when a light-weight component overlaps a heavy-weight component, the heavy-weight component is always on top, regardless of the relative z-order of the two components.

The final point (on overlapping) is the one that is of the most concern to the design of the tool. Placing the web browser object in a JInternalFrame allows it to be moved around the desktop, in the same way as the results frame and other frames on the desktop, and is highly ideal given the desktop workspace being utilized as the focus of the tool. However when the Web browser object is placed into a JInternalFrame, a problem of overlapping between different frames on the desktop occurs, where the web browser's contents will always be on top obscuring all other Internal frames. The problem is illustrated in Figure 5.69. This problem exists because of the fact that a light-weight component re-uses the screen real estate of its nearest heavy-weight ancestor and is therefore restricted to the z-order position of that ancestor. When placing a light-weight object 'on top' of a heavy-weight object, it still appears as if the heavy-weight object is on top, because the light-weight object it is rendering its contents in the native window of the frame (its first heavy-weight ancestor), while the heavy-weight object is rendering its contents in its own native window, which is actually a child of the frame's native window.

The recommended solution to this problem (as stated by Sun Microsystems [33]) is *"Do not place heavy-weight (AWT) components inside a JInternalFrame"*. This is problematic given the aim of integrating the web browser into the desktop and left three choices: (1) place the web browser in an external window (JFrame) where the view will not overlap with other objects, (2) leave the web browser within an internal frame on the desktop and ignore the visual problems of the overlapping frames, or (3) integrate the web browser into the tool, but not within a flexible internal frame, instead as a panel with a fixed position.

As mentioned previously, the external window was not ideal given the fact that users should not have to switch between the tool and an external browser window. It was also decided that the

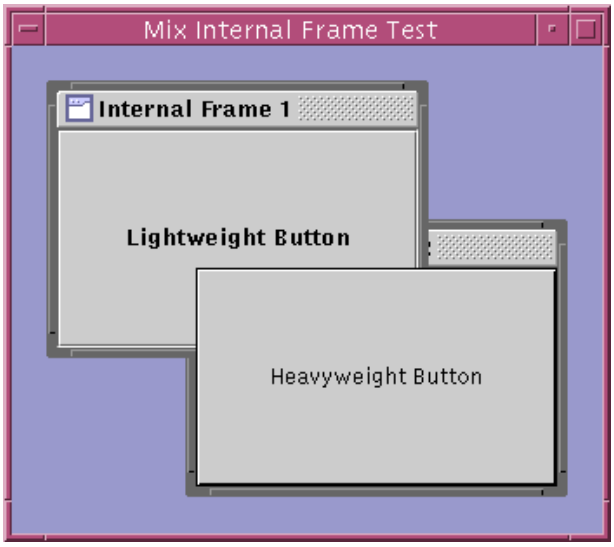

Figure 5.69: **Java light-weight/heavy-weight problems.** A light-weight component is overlapped by a heavy-weight component

problem of overlap was both a visual eyesore and interfered with a user's workspace management, since web browser windows would have to be constantly moved to allow the viewing of search results. In the end, it was decided to split the space between the web browser and the desktop to stop the overlapping of objects. This required the use of a JSplitPane, which is discussed in the implementation of the final version of the EvoBerry interface (see section 5.3).

Speed

In order for the EvoBerry interface to be efficient, it was necessary that all the functions worked quickly and efficiently. While the Java components used to display the interface were indeed processor intensive, it did little to slow down or diminish user's browsing. However, data gathering for the EvoBerry interface presented several challenges because of the long lengths of time needed to gather information across the WWW. A quirk in the Google API is that it returns search results in sets of ten results. Hence, obtaining results beyond the first ten results requires the interface to specify a different starting point from which to gather results. This was troublesome because it required the data gathering program to perform five searches in order to gather fifty search results.

It was necessary to gather the web pages for each each search result returned, in order to gather specific metadata about the web page (e.g. keyword and image counts). This constituted a challenge, since the process of extracting a web page's information required the program to open a port to connect to each website, and download the required page. While connecting to a single website merely takes one or two seconds, when attempting to gather information from several websites this often tripled the time taken for the web search.

In the prototype of the Evoberry interface, the LinkTo function allowed users to see which web pages were linked to the current web page being viewed. The problem with this function, was that the underlying method of obtaining this information required an additional web search to be performed. Compared to its sister function, LinkFrom (which displays its data within a few seconds), this function required at least sixty seconds to gather the data, which may put off users from using this function, as well as slowing down search progress overall. In the end, it was decided to remove the LinkTo function, since the study showed that it did not add much to the browsing experience.

The solution to the problem of gathering both search results and web pages quickly was to utilize different processing threads. Naturally, both gathering processes operate sequentially, gathering one set of data first, storing it, and then once the process is finished, gathering the next set of data. By running the processes in parallel, the time taken to perform each search was greatly reduced (from two minutes to just over 10 seconds), because sets of search results, and web pages were all being gathered in parallel using a large number of threads. It should be noted that in the final version of the EvoBerry interface, the switch from the Google API to Yahoo! meant that search results were no longer being returned in sets of ten, and so search results no longer needed to be gathered through the use of multiple threads, although threads were still used to parse the web pages.

5.3 Final version

Both the prototype and the final version of EvoBerry shared the same essential technologies, both were written using Java (1.4) and both utilized the JDIC web browser to visualize web pages. A minor difference in technologies was a change in the search engine used to gather search results, from Google to Yahoo!. This was partly due to the increase in API functionality that had become available since this research started, and partly due to technical difficulties that had been experienced with the Google Web API (see technical discussion section). The design differences between the two tools, are presented.

A small change in the underlying data gathering systems was made, moving from Google's web service to Yahoo!'s. Essentially the same data was returned, albeit via a different search engine. The display of search results underwent some major changes because it was determined that the bar visualization utilized in the prototype was initially confusing to users. As a result, the display was changed back to the textual representation utilized in the original visual-bracketing technique (see chapter 4). However the bar visualization was still useful as a secondary display technique, and was utilized in the results overview (see section 5.3.1). With the move away from a bar visualization, it became difficult to visualize the term count for each

search result. Combined with problems previously mentioned of establishing relevance through term count, it was decided that the term searcher tool would not be re-implemented for the final version.

The design of frames on the desktop underwent some major changes. Users were spending too much time moving between linked, but physically separate tools, e.g. the tool bar and results frame. This greatly increased user's cognitive load, and it was believed that by more closely associating the controls to the data (integrating the functions and data within a single results frame), a quicker and more efficient form of browsing could be facilitated. It was also decided that the multiple tool frames which existed on the desktop added to the visual clutter, since no more than one tool was ever active at any one time. The tools were integrated into a single frame, allowing better user control. This reduced the visual clutter, and also grouped the tools together so that they could be found more easily.

Both the LinkTo and LinkFrom frames provide the ability to move between the links to and from a web page and were useful for navigating and discovering new information. However the implementation for LinkTo required extra searches to performed, unnecessarily slowing down the browsing process. The decision was made to not to include these functions in the final version.

Both the thread tool and the search type options (url, title and phrase) were both removed in the second prototype. The thread tool was considered obsolete, given the integration of all the separate frames associated with a single thread into a single results frame. The change in search engines created a change in available services, hence there was not enough time to re-implement the search bar functionality, and as a result it was not included in the final version.

Perhaps the biggest change between the two versions was the move from an external web browser to an internal web browser. This was a because of the unnecessary amount of cognitive overhead that was caused by having to switch between the external browser and the work space. While this web browser view could not be fully integrated into a frame on the desktop due to technical constraints, it was however incorporated into a fixed resizable partition on the desktop (using a JSplitPane). This allows users to assign working space to the web browser and work space dynamically as needed. Furthermore tabbed panes are included (like the Opera and Fire Fox web browsers) to aid users in managing web pages.

The final version of the EvoBerry interface can be seen in Figure 5.70. While the tool still shares the same desktop workspace, many of the principal components have been redesigned. The toolbars have now been integrated into the results frames, essentially merging the functions and data related to a single thread, into one single place. The query tool has been moved the to the top of the interface, but has been significantly cut down to its bare components (removal of thread tool, and different search options). All the different tool frames have been integrated into

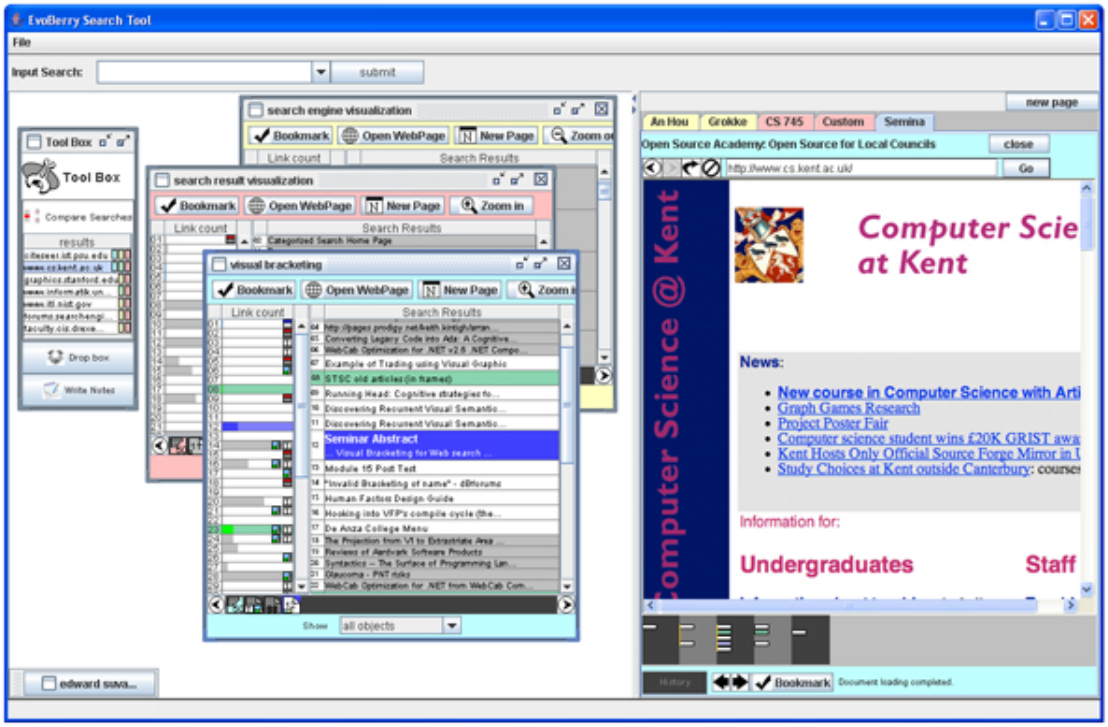

Figure 5.70: **EvoBerry final version.** This is a view of the final EvoBerry version. As with the prototype, this utilizes a desktop interface, however this now shares the main view with a slide-able panel containing a tabbed web browser (far right). The desktop, now only contains two types of frames, the *toolbox frame* (far left) and the *results frames* (three frames in the centre).

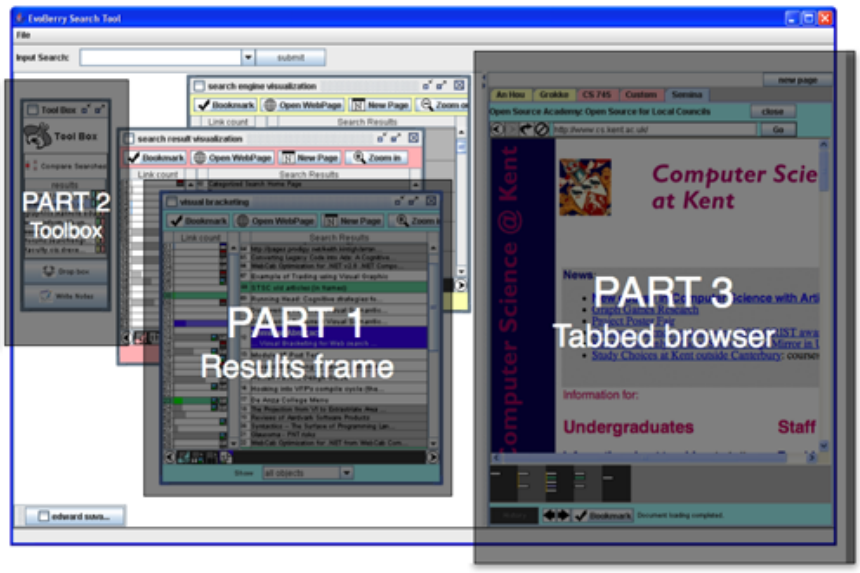

Figure 5.71: **EvoBerry parts 1-3.** The EvoBerry tool is discussed in terms of its different parts/components, (1) the results frame, (2) the toolbox, and (3) the tabbed browser.

a single toolbox frame. The most significant difference between the two interfaces is the new *tabbed browser pane*, which renders the web pages. The following section describes the different techniques and tools used in the final version of EvoBerry. The tool is described in terms of its principle components. There are three main parts (as seen in Figure 5.71) are, (I) the *results frame*, (II) the *toolbox*, and (III) the *tabbed web browser*,

5.3.1 Part I : results frame

As with the previous version, a results frame is returned whenever users perform a search, and is automatically assigned a unique colour. However, the new version of the interface integrates the (previously separate) toolbar functions with the main view of the results as well as the session history. Figure 5.72 shows the new results frame design. The results frame can be split into three view panels, (as shown by Figure 5.73) and a function toolbar across the top of the frame. The search result information returned from the search engine is shown in both the *overview* and the *main view*. Each of these functions is presented.

The toolbar is positioned across the top of the results frame. This contains the four functions that can be used to interact with the data, bookmark, open web page, zoom (in/out) and create new (blank) web page. The overview is shown as a column, located on the left of the results frame. The information returned by the search engine is shown in two views, the overview and the main view. The overview displays the search result data at a lower level of detail than the main view. Each search result in the overview is represented by a bar denoting the number of external links present in that web page. Icons are also assigned to bars which were linked to search results which were not HTML web pages (e.g. pdf, word documents) as well as web pages with special attributes (e.g. lots of images or text).

The main view is located in the centre/right of the results frame, and occupies the bulk of the results frames view. The search result information returned by the search engine is also displayed in the main view. Results are shown as lines of text (the web page's title) and visual-bracketing is applied to the results display. Users can browse the results through several interaction methods including mouse clicks, scrolling, using the mouse wheel, and keyboard key strokes.

The Session bar is shown as a bar across the bottom of the results frame. This is effectively a graphical representation of the prototype's search session history, but instead of adding entries to a drop-down box, the tool adds icons (representing each session) onto the session bar every time a search result is opened.

These different parts of the results frame interact together in different ways, for example, pressing the 'open web page' button in the tool bar, opens the web page associated with the current selected search result in the tabbed web browser, as well as adding a new session icon to

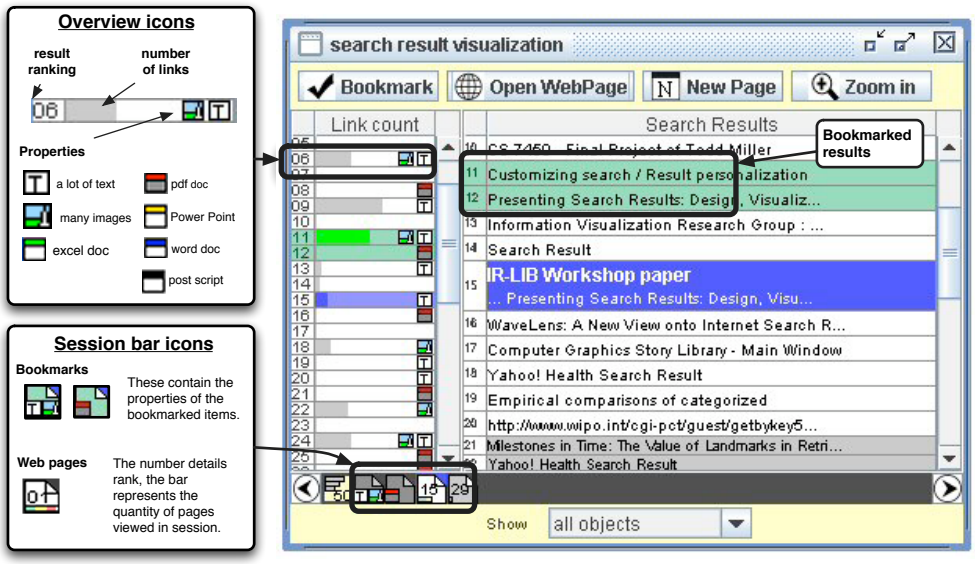

Figure 5.72: **Detailed look at the results frame.** When a search is performed, the search results are returned in a results frame. Each results frame is assigned a unique *thread colour* when it is generated, which is used to associate objects and data that all belong to the same search thread. Along the top is the *toolbar*, which can be used to interact with and manipulate the search results in the main view. The overview (left) shows the search result data at a lower semantic level, and main view (centre/right) shows the results returned using the visual-bracketing technique. At the bottom is the session bar, which contains icons representing each of the search sessions.

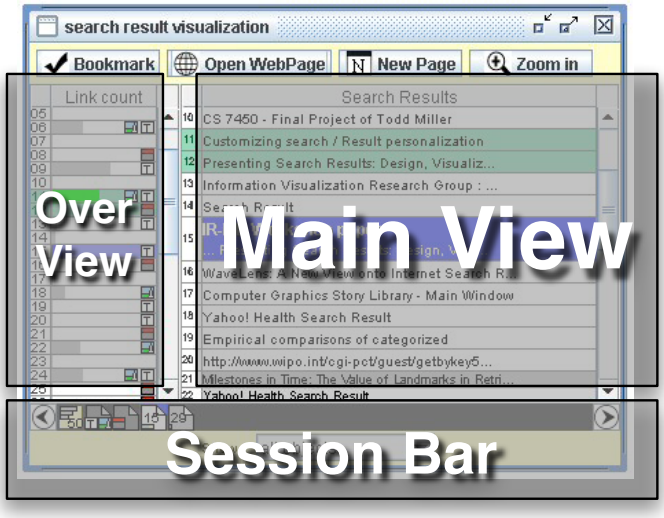

Figure 5.73: **Layout of the results frame.** The three main parts of the results frame are illustrated here, the *over view* (left), the *main view* (centre/right), and the *session bar* (bottom).

the session bar. Descriptions of these interactions are provided in both this section (subsection 5.3.1) and the toolbox (subsection 5.3.2) and tabbed browser (subsection 5.3.3) sections. Next, the four different parts of the results frame (described above) will be described in detail, and discussed.

Toolbar

There are four buttons arrayed across the top of the toolbar: *bookmark*, *open web page*, *new page*, and *zoom in*. Most of these functions replicate previously designed functions within the first prototype, for example, the bookmark button still highlights a search result in green and the zoom button still increases the textual detail available in the main view, and the open web page button still opens the currently selected search result in a web browser. As with the first prototype, both the bookmark and zoom actions can be reversed by pressing the button a second time, while selecting the appropriate result. However there are some minor differences. Because of the changes in interface design, web pages are now opened in the internal tabbed browser view (see tabbed browser section) and zooming in will increase the level of detail of all the results in the main view (not just the selected result). A new function is the new page button, which allows users to create a blank web page in the tabbed browser view, in case they have need of a separate web page.

Overview

The overview's relation to the main view is much akin to the relationship between the view of the aforementioned coordinated bracketing tool as detailed in chapter 4 (see Figures 5.72 and 5.73). In the coordinated bracketing tool, three different coordinated views were used to show data at different semantic levels (i.e. using visual-bracketing). Similarly, in this implementation of the results frame, the overview shows the data in the main view at a different semantic level, and is also coordinated with the main view. This means that when a result is selected in the main view (bringing it into focus and highlighting it in blue), the corresponding result in the overview will also be brought into view. This also works in reverse: selecting a result in the overview will bring the corresponding result in the main view into focus. Some search results in the overview contain icons which are used to communicate the different properties of their associated web pages. This includes alerting users to search results that link to non-HTML data (e.g. pdf and word documents) as well as web pages that meet specific criteria (e.g. web pages containing a lot of text or images.). This helps users identify special properties of search results without having to view them, and helps users seek results of interest. A full list of the overview icons is shown in Figure 5.72.

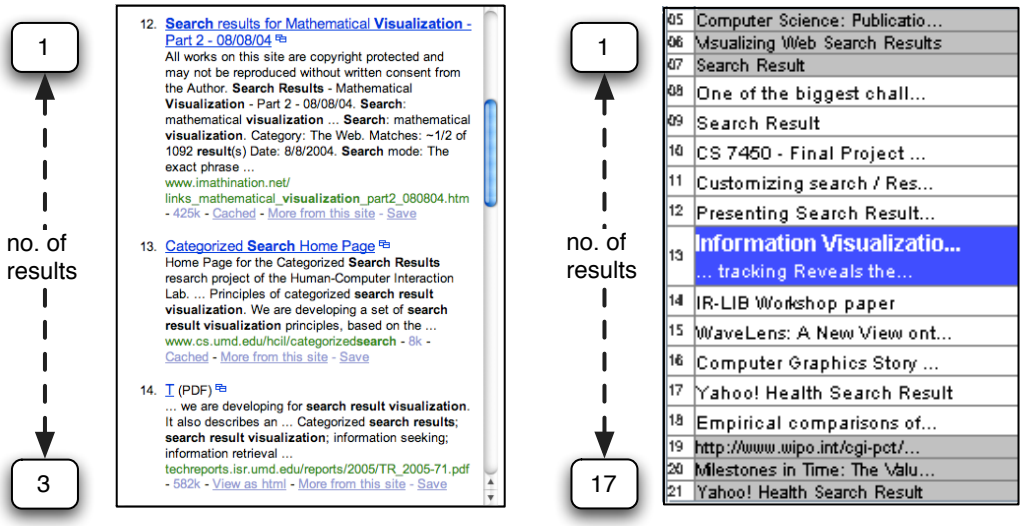

Figure 5.74: **The visual-bracketing technique.** On the left is a view of traditional search result visualization where three results are displayed. On the right is the data data using the visual-bracketing technique, where three times the amount of results can be displayed. The visual-bracketing technique displays different semantic levels of data in 'brackets' (see chapter 4 for full description).

Main view

The main view displays results according to the visual-bracketing method. The display creates three 'brackets' of information. Information in the focus forms the first bracket (currently being selected - highlighted in blue) and shows the most amount of textual. The next bracket is formed by the five results either side of the focus (coloured in white), which show search results at a lower level of semantic detail. The final bracket is formed by the remaining results (coloured in grey), which are shown at a lower font size. A diagram of the visual-bracketing technique applied to the search results can be seen in Figure 5.74. As mentioned previously, a certain amount of coordination takes place between the overview and the main view. However, coordination between the two views (as well as other views) can be affected by other tools. For example, bookmarking a result in the main view (which highlights the result in green) will also bookmark the corresponding result in the overview, and vice versa. Results in the main view are also coordinated with data from other results frames, through the use of similarity highlighting. Like in the prototype's results frame, selecting a search result that shares a website with a search result in a second search will automatically highlight and bring that result into view in that second search (see comparison tool for more details).

Interaction with the results in the main view can be achieved in several ways. Users can move among the search results by simply clicking on the result, this will change the view, bringing the selected result into focus. The focus can also be changed by rolling the mouse wheel, or

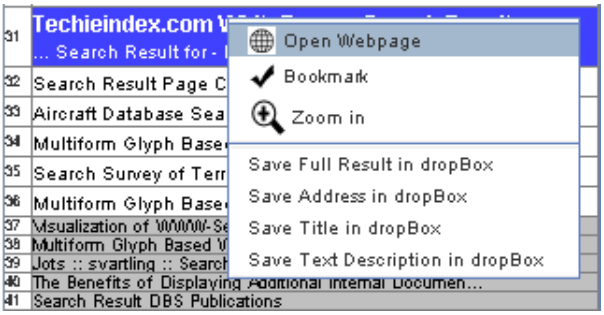

Figure 5.75: **Right-click pop-up.** When users right-click on a search result in the results frame, a pop-up menu will appear, with shortcut functions allowing them to *open a web page* linked to the current search result, as well as *bookmark* and *zoom-in* on the current search result. The pop-up also provides options for sending different types of information to the drop-box.

pressing the up and down keys. The results in the main view can also be manipulated through use of the right mouse button. Right-clicking on a search result will activate a pop-up menu (see Figure 5.75), which will give users quick access to functions in the toolbar (e.g. open web page, bookmark, and zoom), as well as options to add specific data to the drop-box (e.g. address, text, search result). Note that an alternative to having to click the zoom-in button is to double click on the currently selected result, which will automatically increase the level of detail. Double clicking on the result when it is already zoomed-in will open the result's web page in the tabbed browser. An alternative way of sending information to the drop-box (as opposed to right-clicking and selecting a command) is to click and 'drag' the search result from the main view, into the drop box. This automatically adds the whole search result to the drop-box, so that it can be recalled later.

Session bar

The session bar (see Figure 5.73) occupies the same role as the session history in the prototype, keeping a record of all the search sessions created by users, so that they can be recalled later. The only difference is that the sessions are now stored as icons instead of title text. There are two types of icons which are added to the session bar: (1) session icons and (2) bookmark icons. Whenever a search result is opened (and a search session is created), then a session icon is added to the session bar. The different session icons can be seen in Figure 5.72. In general, each icon is represented by a small picture of a white piece of paper, and contains a number in its centre, and a small bar across the bottom of the icon. The number represents the ranking of the search result which was used to create the search session, and the bar indicates how many web pages were visited during that session. Notice that the bar has three different colours, green indicates 1-5 pages, yellow indicates 5-20 pages, and red indicates 20 or above pages. If the particular search result has any special properties or is of a special file type then the icon will also have special symbols attached to it (these are the same symbols used in the overview, see Figure 5.72

for a detailed description).

Whenever a search result is bookmarked in the main view, a bookmark icon is added to the session bar. The purpose of this is to allow users to quickly relocate bookmarks which are outside of the current view. Bookmark icons are designed in exactly the same way as session icons, except for their colour, which is green, not white. Users can interact with session (and bookmark) icons in different ways. Clicking on a icon in the session bar will show the selected icon in full colour (bringing it to the foreground), while 'graying-out' the other icons (sending them to the background), helping users identify which icon is currently being manipulated. Double clicking on a session icon has two effects: (1) if the corresponding session is already open in the tabbed browser, it will bring its tab to the front, and (2) if the corresponding session has already been closed, then it will reopen the session in the tabbed browser. Double clicking on a bookmark icon will change the main view's focus to that specific result. As with the main view, right-click can be used to activate a pop-up menu, with a set of commands, including 'open session', 'copy session' and 'remove session'. The website address of the web page linked to the session icon is displayed at the top of the pop-up menu for easy identification. Users can also identify the icon by hovering the mouse cursor over the session icon, which will in turn pop-up a tag stating the page's title. If users generate more icons on the session bar than there is space, then the scroll buttons (arrowed button on the left and right of the bar) can be used to scroll to the icons not being currently displayed in the view. Alternatively, if there are too many icons, users can choose to filter the icons based on type (bookmark or session), using the filter tool (located below the session bar).

5.3.2 Part II : toolbox

The toolbox frame combines two of the tools of the prototype interface, the *comparison tool* and *dropbox*, and a third tool, the *notepad*, all contained within a single frame on the desktop. Each of these tools can be seen in Figure 5.76, on the left is the initial state showing buttons to activate each of the tools. When one of the tools is activated, it occupies the majority of the frame's space, and a pair of tabs representing the two inactive tools are shown, and can be clicked upon to switch between the tools. Details of each tool are presented.

Comparison tool

The tool mirrors the first prototype version of the comparison tool in functionality. Whenever a search is added to the workspace, the intersection of websites between the new search and the existing searches is calculated, and the resultant websites are added to the comparison view. This allows users to view which websites appear in multiple searches. The tool also shares similar interaction techniques with the prototype, clicking on a website entry in the comparison tool

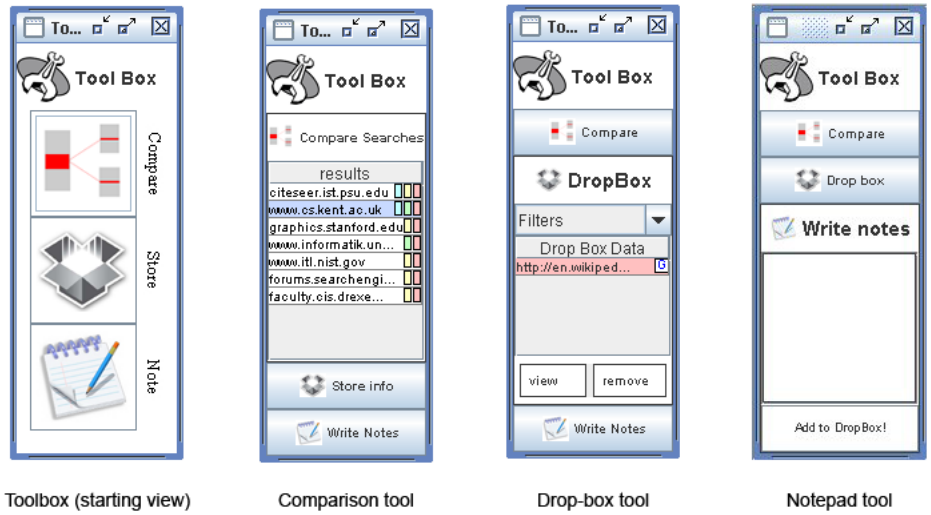

Figure 5.76: **The toolbox and its three different tools.** This figure shows the starting view of the toolbox (far left), and the views of the three different tools, the *comparison tool* (centre-left), the *drop-box tool* (centre-right), and the *note-pad tool* (far right). When EvoBerry is first activated, the toolbox is displayed in its starting view, but when one of the tools is selected, the view changes to that of the tool. The comparison tool is used to visualize intersections between the multiple searches, the drop-box allows users to add search results for storage, and the note pad allows users to write textual notes (that can be stored in the dropbox).

will highlight its corresponding search results in their results frames, as well as bring them into focus. Double-clicking on a website entry will open it in the tabbed web browser. The major difference between the prototype version, and final version is the use of display techniques. In the prototype, each web page entry was simply represented by the website's URL. In the final EvoBerry version, coloured blocks show which searches the website is found in. The number of blocks indicates the quantity of searches that this URL exists in, and the colour of the blocks associates the URL with the search thread that it came from.

Drop-box

The drop-box also shares similar features with its prototype predecessor. Many of the basic interaction techniques are similar; results can be added to the drop-box, by either (1) dragging them from the results frame directly into the dropbox, or (2) by right-clicking on the result in focus and selecting which piece of information to send to the drop box. Double-clicking on drop-box entries will reopen the information in small frame on the desktop. Display techniques are also similar, drop-box entries are represented by a string of text (the URL, but varies depending on the type of information stored), and an icon which represents what type of information is contained in the entry. Results are also colour coded by which search thread the information came from, and can be filtered according to type. However this implementation of the drop-box differs from the prototype. Two buttons have been added to the tool, 'view' and 'remove'.

The view button replicates the double-click function (reopening the selected information), but provides a more obvious visual cue. The remove button will remove the selected entry from the drop-box, a function that can also be performed by selecting and dragging an entry outside of the drop-box. The notepad function was replaced and extended in the final version with the notepad tool.

Notepad

The notepad is a simple tool consisting of a text box, in which users can write notes, and a button which submits the notes as an entry in the dropbox.

5.3.3 Part III: Tabbed Browser

When redesigning the EvoBerry prototype, one of the main concerns was that using an external web browser increased user's cognitive overhead, because of the amount of switching that had to be done between the web browser and Evoberry views. The ideal solution, would be to include the web browser within its own internal frame, so that it could be viewed side-by-side with the results frames, and could be repositioned and resized with ease. However there were some technical constraints as to how the web browser could be displayed in Java. The problem stemmed from the fact that the web browser (a Java heavy-weight component) could not be added to a frame on the desktop (a Java light-weight component). Thus the tabbed browser was added to the main view inside a split-pane (Java JSplitPane).

The split-pane (see Figure 5.77), can be resized along the horizontal axis by dragging the edge of the split-pane across the main view. In this implementation the split pane can be slid across the rest of the desktop covering the other frames and obscuring the background. The advantage of the split-pane, is that the web browser is no longer external. It can now be used side-by-side with the results frames, and can be resized as necessary, depending on whether users are concentrating on the web browser or the results frames more. The disadvantage of this method is that it is very easy to lose frames 'underneath' the split-pane if users are not careful (see Figure 5.77). Given that the alternative to the split-pane was an external browser, it was decided to utilize the split-pane in this way. The layout of the tabbed browser can see in Figure 5.78. The tool clearly consists of three main parts: the browser controls (top), the web page view (middle), and the visual-history toolbar (bottom). These are be discussed in turn.

Browser controls

As with the prototype's version, the web browser contains all the standard web browser functions. Buttons are provided for the backwards, forwards, refresh, and stop functions. An address bar is also provided, which allows users to see the URL address of the web page currently being

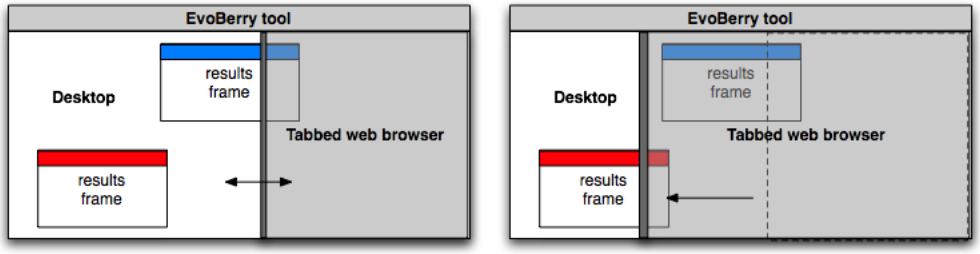

Figure 5.77: **The Splitpane.** This figure illustrates usage of the splitpane. On the left the splitpane shares an equal amount of screen space between the desktop and the tabbed web browser. Note that the tabbed web browser is shown to be transparent to illustrate the fact that objects on the desktop are obscured by the web browser view when it is moved. The advantage of obscuring objects (as opposed to repositioning them) is that users frames and desktop space will not become distorted when the tabbed web browser view is extended. However the disadvantage of this can be seen on the right, where moving the web browser view too far may cause frames to be lost underneath.

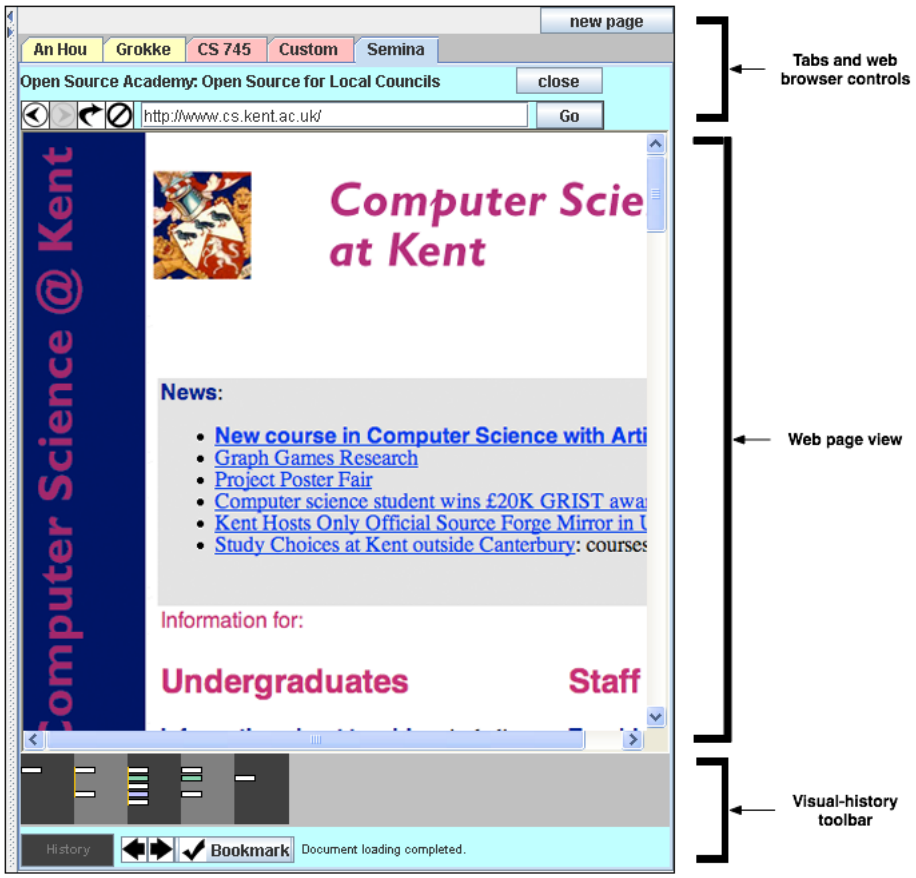

Figure 5.78: **The tabbed web browser view.** This shows a view of the tabbed web browser used in the EvoBerry interface. The web browser is broken up into three views, the browser controls (at the top), the main web page view (centre), and the visual-history toolbar (bottom).

viewed, and modify the URL. However, unlike the external web browser, this web browser utilizes a tabbed presentation. When users open a search result in the tabbed browser, a search session is created within a new tab. The tab will, from that point forward, contain all the information on web pages viewed in that specific tab and record it in the visual-history toolbar (see below). When a second search session is created, a second tab is created, and immediately comes to the foreground.

Switching between search sessions is simply a matter of clicking on the corresponding tab. Each tab can be identified in one of two ways, either (1) by its thread colour, where search sessions from the same search thread are assigned the same colour, or (2) by a label assigned to each tab, which is based on the first unique word in the search result description. As mentioned previously, when a search session is created, a session icon is added to the session bar of the result frame where it originated from. This means that whenever the tab containing the session is closed, it can be reopened, simply by double-clicking on the session icon. Additionally, all the web pages viewed in that session can be recalled through use of the visual-history bar.

Web page view

As with the prototype, the design is developed around the JDIC web browser [23], which in turn renders web pages using Microsoft's Internet Explorer [90]. A quirky feature of this method is that many of the Internet Explorer's shortcut functions are active, even though their equivalent buttons are not visible, for example, pressing the keys CTRL and F will pop-up the Internet Explorer's 'find term' function.

Figure 5.79: **The visual-history toolbar.** This view utilizes the *visual-history* technique explained in chapter 4. At the top of this view, is main view of the visual-history toolbar, that visualizes users web page history as a two-dimensional map. Below are buttons which can be used to interact with the view, allowing users to *hide* the visual-history, *scroll* the results, or *bookmark* specific web pages.

Visual-history toolbar

The visual-history tool bar (see Figure 5.79) operates in much the same way as its prototype predecessor. Both share the same column/block layout and utilize the same visual-history techniques to deal with web page block placement and overflow (see section 4). Both the

prototype and the current visual-history toolbar utilize similar interaction techniques. When a new web page is opened, it is added as a block in the display, and users can revisit any page in the history simply by clicking on its corresponding block. A blue coloured highlighting allows users to keep track of which block (web page) is currently being viewed. However, there are a few differences between this visual-history toolbar and the previous prototype. A bookmark function has been introduced, that allows users to highlight web pages of interest within the visual-history toolbar. This function highlights specified blocks in green, and can be reversed simply by clicking the button a second time. Also the overview bar has been replaced with a pair of scroll buttons, that allow users to scroll the visual-history display when the columns extend past the width of the browser.

5.3.4 Overview

This section presented the final version of the EvoBerry interface. Design changes have taken place between the development of the prototype interface and the final version. The three most significant changes to the interface were (1) the integration of tools and data into a single results frame for each search thread, (2) a change in the results frame display, from graphical bars to text, and (3) the move from a external web browser to an internal tabbed web browser. Each of the tools, developed as part of the interface, contributed to aiding users in performing evolving searches. Each of the different interface issues that the tools addressed is presented.

The issues of information visualization and opportunistic searching were both addressed through the use of the visual-bracketing and visual-history tools. The application of the visual-bracketing technique to the search results within the results frame allows users to view more results, at varying levels of semantic detail and provides a graphical representation which promotes opportunistic movement through the data. Similarly the graphical display of the visual-history tool provides a more compact presentation of the information (web page history in this case), as well as displaying the list of search result data in a manner that is more easily browsed, allowing users to move between search results opportunistically.

The issue of information recall was addressed through different automatic and manual history systems. Search sessions are automatically recorded as session icons on the session bar, which allows search sessions to be recalled with ease. Once a search session is reopened in the tabbed browser, that session's visual-history toolbar will be displayed, allowing users to recall specific web pages from the history instantly. The bookmark function can mark specific search results to be recalled later. Search results can also be dragged into the drop-box and reopened later.

The issue of information-seeking was addressed by the comparison tool, as well as other minor tools provided for finding specific information, both in the search result data and history data. The comparison tool (in conjunction with the similarity highlighting) allowed users to view the

intersections of multiple searches, in order to find interesting results. Tools were provided to filter the search history data, in both the drop-box, and the session-bar.

The issue of information management was addressed in several ways. A desktop interface allowed users to better manage the tools and data within the workspace. Data management is present at two levels (1) data is automatically managed into threads (using thread coloring), sessions (via the session bar) and web pages (within the visual-history toolbar), and (2) data can also be manually managed through use of the drop-box.

Chapter 6

Experiment and results

Experimentation is an important part of any research. After thoroughly investigating both the design and implementation of an evolving search tool, it is important to critically analyze the contributions of the tool. However, evaluating visualizations is difficult; experiments with human participants are time consuming, defining and setting up the whole experiment is a challenge on its own, and it is often hard to analyze and consolidate any certain conclusions.

As part of the development of the EvoBerry interface, it was decided that an evaluation must be performed to determine the effectiveness of the tool in aiding users to perform evolving searches. It was hypothesized that users of the EvoBerry interface would exceed the performance of users of traditional web search tools in terms of speed, accuracy, and number of searches generated (this is elaborated on later). This chapter is split into three main sections. The first section, *visualization experiments*, discusses the design of different visualization experiments in the research. The second section introduces the concept of the *information-seeking experiment*, and how these are different to previous experiments in the area of search result visualization and information retrieval. The third section, the *experimental design*, details the design of the experiment used to test the EvoBerry interface.

6.1 Visualization experiments

Most visualization experiments can be classified as either *Heuristic Evaluations*, which are concerned with finding problems with the tool's usability, or *Empirical Evaluations*, where the efficiency of a tool is determined, based on observations of how users interact with the tool, and a statistical analysis of the their performance. The differences between the two forms of evaluation are summarized below:

Heuristic evaluation belongs to a set of techniques called *discount usability engineering methods* [81], which emphasize a quick, immediate and easy way to evaluate user interface designs.

Jakob Nielsen's usability heuristics	
Visibility of system status	Does the system keep the user up-to-date?
Match between system and the real world	Is information presented in a logical order, with words, phrases and concepts familiar to the user?
User control and freedom	Does the system support 'undo' and 'redo' functions on user actions?
Consistency and standards	Are actions and labels consistent through out the tool ?
Error prevention	Have all errors been eliminated?
Recognition rather than recall	Are the objects, actions and options visible? Are all tools and buttons clearly defined and visible?
Flexibility and efficiency of use	Does the tool cater for experienced as well as novice users? Does the tool offer customizability or shortcuts to speed the interactions of more expert users?
Aesthetic and minimalist design	Has all irrelevant information been removed ? Are all the functions and objects free of unnecessary visual clutter?
Help users recognize, diagnose, and recover from errors	Are error messages provided? And if so are they helpful?
Help and documentation	Is documentation provided where necessary?

Table 6.12: **Nielsen's usability heuristics.** Nielsen [82] outlined ten rules for good usability, his Usability Heuristics. These can be used as part of a heuristic evaluation, to identify which parts of a interface will cause the user problems.

Features	**Heuristic evaluation**	**Empirical evaluation**
Objectives	Evaluate the tool's usability.	Varies. Often to evaluate the effectiveness of the visualization technique or tool.
Methodology	Experts inspect tool with a set criteria.	Observe subjects using tool in strict experimental conditions.
Structure of experiment	Single condition (the tool itself) is tested.	Can vary. Usually either one or two conditions are tested, although several more conditions is not uncommon.
Data obtained	Qualitative data. Usually problems with the interface.	Both qualitative and quantitative data. Details about users satisfaction, as well as statistics about user performance.
Turnover	Quick. Several iterations are possible, and results can be processed quickly.	Long. Often takes a long time to conduct several experiments.
Amount of personnel	Small. 3 to 4 is optimal.	Can vary between 5 and 50. The more personnel, the more accurate the results.
Personnel	Usability specialists.	Test subjects. Backgrounds and experience vary depending on the objectives of the experiment.

Table 6.13: **The differences between heuristic and empirical evaluations.** This is a summary of the main differences between the two different forms of evaluating visualizations.

The objective of this form of evaluation is to identify flaws in the tool's design before it is empirically evaluated. To this end a handful of usability experts are recruited to evaluate the visualization using a set of established criteria. The usability experts identify flaws in the visualization, and communicate these to the tool designer who can re-design according to the suggestions made. This form of evaluation is quick and easy, and is best repeated several times. This method has come under criticism because of the lack of statistical evidence to support any claims of the tool's effectiveness, and as a result heuristic evaluations are usually a precursor to a more detailed empirical evaluation.

Jakob Nielsen [82] outlined a set of ten usability heuristics (see Table 6.12), which became associated with Heuristic Evaluation. The concept was to provide usability evaluators with these set of criteria when inspecting the visualization, so that they may appraise the visualization based on the ten criteria. For each criterion, the evaluator would assign a severity rating (usually between one and ten) which would indicate where any potential visualization flaws may lie. Nielsen [82] commented that the optimal number of evaluators is four (see Table 6.13), and also showed that effectiveness (in terms of benefits compared to cost) of the evaluation dropped as the number of evaluators increased. A pseudo-heuristic evaluation, based on Nielsen's usability heuristics, was performed as part of the design process of the EvoBerry tool with two usability experts. While the number of evaluators was not adequate, the comments provided were helpful, and several iterative evaluations were undertaken before the tool was finally completed.

By comparison, empirical evaluations are more stringent and rigorous. While the specific objectives of empirical evaluations can vary from experiment to experiment, they generally revolve around testing the effectiveness of a visualization design or tool. Empirical evaluations usually take the form of a strictly controlled experiment, where user's interactions with the tool are observed and recorded. To this end, a group of test subjects are gathered (numbers can vary between 5 and 50) and are usually presented with a set of tasks to undertake within a set time. The nature of these tasks can vary greatly depending on the experiment, although fact-finding and comparison questions are common. Sometimes users are asked to perform specific tool interactions (e.g. bookmark page X). Data from empirical evaluations can be both qualitative and quantitative. The qualitative data is usually based on information gathered from questionnaires (given either before or after interactions with the tool), and is used to obtain personal information as well as opinions about the interface from the user. Quantitative data usually comes in the form of speed (time taken to complete tasks) and accuracy (number of correctly answered questions), but can vary depending on the system in the experiment, e.g. a search result experiment might also capture data on the number of searches generated and the number of web pages looked at. This data can be analyzed using statistical analysis to determine trends in user behaviour and tool effectiveness.

Unlike heuristic evaluations, these take a long time because larger numbers of people are involved. One of the biggest differences between these two forms of evaluation is their structure. The heuristic evaluation only involves testing a single visualization (a single test condition), comparing it against a set of established criteria. However an empirical evaluation can compare data from more than one visualization, and hence can test several different conditions. For example, in an experiment by Hightower et al [47], the effectiveness of the Pad++ navigation tool was tested, and two conditions were used: web browsing with the Netscape Navigator web browser only, and navigating with the Netscape Navigator and Pad++ interface.

When an experiment tests more than one condition, it will generally fall into one of two designs, either the *between-subjects* design, where two separate sets of subjects are used for each condition, or the *within-subjects* design where all the subjects are exposed to both conditions. Each design has different advantages and disadvantages.

The between-subjects design allow the experimental designer to keep the design constant between the different conditions; for example, the task "find a web page containing a picture of a kangaroo" can be applied to different conditions, since different subjects are used for each condition. However, in a within-subjects design, where both conditions use the same subjects, users would know where to seek out their answers after undergoing one condition. However the problem with the between-subjects design, is that larger numbers of subjects are needed, since each subject has only one condition applied to them. It is more difficult to compare sets of users, since differences in subject's abilities may make results incomparable.

In a within-subjects design, the differences in subject backgrounds and abilities can be controlled because both treatments are applied to the same set of users, making their results comparable. Because both treatments are applied to the same users, a smaller test group is needed. However, the experimental designer must be careful to craft two sets of tasks that are similar in design, but do not give users the benefit of any learning effects that may take place (which would aid them in completing the tasks). To this extent, experiment designers can counterbalance the experiments to reduce the learning effects, for example with when comparing two interfaces (A & B) with two sets of tasks (T1 & T2), the experimental designer would split the group into two randomly generated groups, where one group would use Task T1 with interface A, and Task T2 with interface B, and the second group would use the opposite.

Perhaps the biggest difference between empirical evaluations and heuristic evaluations are the personnel involved. Heuristic evaluations only require a handful of subjects for evaluation, whereas empirical evaluations need between 5 and 50 people (the more the better). These numbers generally depend on factors such as user background requirements, time taken to complete experiments, and availability of subjects. The test-subjects chosen for the experiment can also

vary, based on the specific requirements of the experiment, e.g. North et al. [83] specifically selected users with a computer science background and knowledge of database manipulation when testing their Snap-together visualization. Motivation can also play a part in most experiments; users are often given an incentive such as money, or course credits in the case of students, in order to get them to take part in the experiment. Some experiments are specifically designed to motivate the user to perform well, by offering performance based rewards (as seen in Veerasamy et al. [117]).

It should be noted that an empirical evaluation is sometimes preceded by a usability evaluation, in order to determine whether the designs and techniques used in a tool will be understood by users. Unlike heuristic evaluations, these are performed with normal experimental subjects (not experts) and usually require larger test subject populations. Examples include the experiments of Grewal et al. [38], North and Shneiderman [84], and Nowell et al. [85].

Grewal et al. [38] wished to test the viability of their R-wheel and Tepee visualization designs before implementing them in a tool (and consequently performing an empirical evaluation). These novel visualizations displayed the search result information from multiple-term queries. In their experiment, users were asked to demonstrate their comprehension of the visualization by ranking presented images of the visualizations, according to their perceived relevance. By ranking the visualizations accurately, the users displayed a good grasp of the concepts behind the visualization. The design of the experiment was distinctive, since no tool was involved in the experiment (simply static images of the search result representation), and hence none of the user's interactions with the visualization were observed. Grewal's intention was merely to test the viability of the design before investing time in placing either design into a tool. As a result, there was no need to compare the visualization against a search result visualization, success was determined on the user's comprehension of the visualization and the principles which govern it. Similarly, an experiment by North and Shneiderman [84] tested the abilities of users to construct their own coordinated-visualization interfaces, as a prelude to testing their operation with their coordinated views tool. In the experiment, subjects were given specifications (either as an image or textual) from which they were meant to build a coordinated interface using North's Snap-together tool. Like Grewal's experiment, the purpose was to determine user's abilities to comprehend the new visualization principles introduced as well as manipulate them for more effective interactions. However, unlike Grewal's experiment, the subject was told to interact with the tool, although the operations that the subject were asked to undertake did not fully simulate typical user interactions with the tool.

The differences between these two models is elaborated upon in this section, which in turn

takes a more detailed look at the differences between the information retrieval and information-seeking experiments, as well as the reasons for these differences and the challenges in designing an information-seeking experiment.

6.2 Information-seeking experiment

The research literature details experiments that have been undertaken to measure the effectiveness of different search results visualizations. However, the majority of these are classified as *information retrieval* experiments, and are unsuitable for testing evolving searches. In this work, a distinction is made between information retrieval and information-seeking experiments. The differences between the two forms of experiments, reflect changes in search technology, and users search behaviour, which in turn have forced new testing requirements for search visualizations in online environments.

Degree of control

The different types of experiment impose different degrees of control over the behaviour of users during a search. Information retrieval experiments often exercise a high degree of control over the way users behave during a search, e.g. controlling the search terms they use, the data they view, and reformulation patterns. In contrast, information-seeking experiments often exercise a low degree of control over the way users behave during a search, in order to promote a more natural search process.

The majority of information retrieval experiments restrict the information available to the user, for example, Hightower et al. [47] restricted their test subjects to only viewing information from the CHI database and National Park Service website during the experiment. Similarly, in an experiment by Veerasamy et al. [116] the test subject's data were restricted to a database containing information from the TREC information retrieval track. The advantage of this methodology is that it allows experiment designers to predict the answers to specific experimental tasks, by reducing the variation in the information provided to the user. The disadvantage of this methodology is that the natural web search process of users is affected by restricting movement across the information. For example, users will not be free to explore the information opportunistically, and cannot follow hyperlinks to external sources (that would normally be available) which may in turn lead to relevant information.

Some information retrieval experiments restrict users in their search methodology, affecting their ability and opportunity to reformulate their queries. For example, in experiments by both Reiterer et al. [92] and Sebrecht et al. [99], test subjects were provided with set queries to enter into the search interface, and were not allowed to define their own queries. Reiterer further

constrained test subjects by not allowing them to reformulate their queries. The advantage of this method is that by controlling the search methodology of test subjects, the environmental variables are kept consistent which makes the searches more comparable, and the impact of the visualization more apparent. However the disadvantage of controlling the search methodology is that it disrupts the user's natural search process. This is best illustrated by the fact that removing reformulation from the search process will greatly affect the chances of users finding relevant results if they began with only a vague information need.

Some information retrieval experiments restrict users, in terms of the type of information that can be viewed. For example, in an experiment by Veerasamy and Heikes [117], test subjects were forbidden from viewing web pages during an experiment. The purpose of this was to make users evaluate the relevance of web pages "blindly" (without being able to first view the web page) and thus force them to rely solely on the information provided by the visualization. The advantage of this methodology, was that it evaluates the visualization's effectiveness without the impact of external influences. However, the disadvantage of this methodology is that very often users in an evolving search collect information from viewing web pages to reformulate their queries, thus by restricting them from viewing web pages, the user's natural search process is disrupted and search effectiveness is reduced.

Tasks

Different types of experiment also use different sets of tasks. Information retrieval experiments often use directed search tasks, which are often quick, easy to perform, and can usually be satisfied with a single query. Information-seeking experiments often use more difficult tasks, which cannot be satisfied with a single query, such as tasks requiring the comparison of information from several different sources. A consequence of using different tasks is that the different experiment types also have different durations, information-seeking tasks often taking longer because of increased difficulties of the task. The reasons for these differences stem from the models that the different experiments are based on. Information retrieval experiment tasks are based around the information retrieval search model, which consisted of directed search strategies, thus information retrieval experiments mostly utilized tasks based on directed searching. Information-seeking experiment tasks are based on information-seeking models, and thus consist of browsing strategies. As a result, information-seeking experiments mostly utilize tasks based on browsing.

A consequence of the difference between experimental tasks, is that the different tasks also have different durations. Information retrieval tasks utilize directed questions and thus are quick to complete, for example, Hightower [47] reported average completion times to be around 60 seconds, and Sebrechts [99] reported task times of no greater than 120 seconds per task.

By comparison, information-seeking tasks often have far longer durations, due to the amount of browsing that users need to undertake in order to gather enough information to form an effective query. For example, Reiterar's extended fact finding tasks [92], and the comparison tasks present in Hightower et al [47], both cannot be satisfied with single queries.

Time restrictions

Restrictions between the different types of experiment can vary. Information retrieval experiments often impose a time limit on each task, and are short in duration, while information-seeking experiments often have no time limit, which often makes their duration more variable. Information retrieval experiments often impose a time limit for each task test subjects undertake, for example Reiterer's [92] fact-finding tasks were limited to five minutes each. By comparison, information-seeking experiments do not restrict the user's time, because (1) the subjects often need to spend a long time browsing and comparing information and (2) restricting the user's time pressures them into completing the tasks quickly, and may discourage them from browsing opportunistically.

The two types of experiments have different advantages and disadvantages. But an information-seeking experiment is more suitable for testing an evolving search interface for the following reasons: (1) information-seeking experiments allow users to retain a more natural search process, unlike information retrieval experiments, which restrict several aspects of the user's search, such as the data and search methodology, and (2) information-seeking experiments allow users to reformulate their searches, an important aspect of the evolving search process. However information-seeking experiments are not easy to design, and are difficult to implement, something which will be described in the next section.

6.2.1 Experimental design challenges

Because information-seeking experiments aim to create a different testing environment to information retrieval experiments, the challenges and difficulties are different. The main difficulty in designing an information-seeking experiment for testing an evolving search interface is to simulate a user's evolving search task accurately. An evolving search task is difficult to complete, it requires users to create multiple searches either because they do not have enough information to formulate an accurate search query, or because they need to gather information from different sources. A result of these multiple searches is an increase in experimental time. Two challenges that can be identified from this description of the evolving search are the need to design a set of 'difficult' search tasks, and the difficulties encountered by the increased experiment time.

A 'difficult' search task can be defined as a one that cannot be completed with a single query,

completed with information from a single website, or completed with user's starting knowledge. The best way to address all three criteria is by using a comparison task. When users are asked to compare different pieces of information, they are forced to draw information from several different sources, which often means generating several different queries, as well as consulting several different websites.

There are many dangers with using comparison tasks; many websites exist that provide comparisons on certain topics (e.g. electronic equipment, geographical facts), and the WWW is home to many encyclopedic web sites (such as Wikipedia.com [120]) which could aid users in quickly finding the required information. Mostly this requires the experiment designer to be diligent and investigate the area thoroughly before setting the experiment. Tasks that involve a visual comparison of the information are usually good for evolving search tasks, because the information cannot easily be found through the use of a textual query (e.g. a comparison of the positions or distances of different locations on a map).

The second challenge is based around the increased experimental time, a result of using difficult search tasks, and of encouraging users to browse. This in turn generates two experimental problems, test subjects are less likely to volunteer for the experiment because the length of time required to complete the experiment, and as time passes, users will eventually tire of the tasks and as a result their performance will deteriorate. Certainly, reducing the experimental time is an important factor in both of these problems, however this must be balanced with keeping the search tasks difficult to complete, as well as retaining the users natural search process.

Looking at the first problem, the traditional monetary reward is often a sufficient incentive to take part in an experiment, as long as the value of the reward compensates the elapsed time equally. The second problem involves motivating users during the experiment which can be more difficult. One idea is to use themed searches that would interest the user, for example questions about planning a holiday in an exotic country. By engaging the attention of the users with interesting content, they are more likely to browse opportunistically.

The problems in designing an information-seeking experiment for testing an evolving search interface have been presented, and solutions have been discussed. The next section presents a description of an evolving search experiment, based on the ideas discussed.

6.3 Experimental design

In order to test the effectiveness of the EvoBerry tool, an information-seeking experiment was designed and executed. The hypothesis for the experiment was that the EvoBerry interface, when compared against a standard web searching tool, would perform significantly better in the following areas:

- **Speed:** Users of the EvoBerry interface will spend less time performing evolving search tasks than users performing the same tasks using standard web search tools.
- **Accuracy:** Users of the EvoBerry interface will answer more of the task questions correctly than users performing the same tasks using standard web search tools.
- **Searches performed:** Users of the EvoBerry interface will perform less web searches during an evolving search task than users performing the same tasks using standard web search tools.

Test subjects would be gathered and instructed in the use of the EvoBerry tool, then test subjects would perform two evolving search scenarios (each comprised of three evolving search tasks), one for each experimental condition. The experimental conditions would be performing the evolving search scenario (1) using the EvoBerry tool, and (2) using a standard web search tool (Microsoft Internet Explorer). These will be referred to as the EvoBerry and Internet Explorer (IE) conditions respectively. The experimental specifics are summarized in Table 6.14. This section describes the experiment in terms of the *experimental procedures* carried out, and the *experimental data* gathered during the experiments.

Experimental specifics	
Conditions being compared:	Two. (1) Using the EvoBerry tool, (2) using Internet Explorer.
Experimental design:	Within-subjects design.
Time limit:	None. Estimated completion time is 30 minutes.

Test subjects

Target user:	Subjects with computer experience who use search engines extensively (at least once every day and have performed evolving searches before).
Test subject population:	20 users.
Monetary incentive:	Performance based. £5 for participating, and an additional £5 for completing the majority of questions correctly.
Training:	Oral presentation of the tool's functions, as well as a small booklet which contained instructions on how the different tools worked. Users were given training thirty minutes time prior to the start of the experiment.

Technology

Search engine utilized:	The Yahoo! search engine was utilized for both the EvoBerry tool and the Internet Explorer tool.
Experiment computer:	The experiment was conducted on a Pentium 4, 2.4 gigahertz machine, with a 19 inch TFT monitor, running the Windows operating system.

Table 6.14: **Experimental specifics.** Note that specific information was obtained on test subjects using a pre-experiment questionnaire, in order to filter out subjects which did not fit the profile of the target user.

Holiday scenario: Thailand	
1.	You decide to go to Thailand on holiday, and you want to spend most of your time on the beach. You haven't settled on which seaside town you want to visit, but you want one with easy access to an airport, so you can take a plane straight from Bangkok airport. *Find 3 airports that are based in coastal or island towns in Thailand.*
2.	You remember that not too long ago there was the tsunami disaster in Thailand which destroyed many beach resorts in Thailand. *Find 3 beach towns in Thailand that were unaffected by the tsunami.*
3.	You find out that there are no flights available to go to Phuket, but you still want to visit the islands in that area. *Find the next closest airport to Phuket airport.*
Holiday scenario: South Africa	
1.	You decide to take a safari Holiday in Africa, because you hear about the "big five", a term used to refer to the 5 most popular safari animals. *Find 3 countries where you can go see the "Big Five" animals.*
2.	You finally settle on visiting the country of South Africa. *Find 3 South African national parks or game reserves, at which you can view the "big five".*
3.	You are unable to get a flight to South Africa, but you find out that several of South Africa's national parks extend beyond the boundaries of South Africa, joining national parks in other countries. *Name 1 of these special national parks, that join national parks in South Africa to national parks in other countries.*

Table 6.15: **Evolving search scenarios.** The two scenario's, *Holiday scenario: Thailand* and *Holiday scenario: South Africa*, were used in the experiment (one for each test condition). Each task consisted of some background information (top - normal text) followed by a question to answer (bottom - italicized text). Each scenario contained three questions.

6.3.1 Experimental procedure

The experiment compares the performance of test subjects in two conditions, when using the EvoBerry tool to complete an evolving search, and when using a standard web search tool to complete an evolving search. Time and monetary constraints restricted the experiment to only 20 test subjects, and as a result a within-subjects experimental design was chosen in order to maximize the significance of the results due to the less than ideal numbers of test subjects. To minimize the impact of any learning effects, counterbalancing was applied (half the users performed the experiment in a different order). In each condition the subjects were presented with a scenario (see Table 6.15) and asked to complete three evolving search tasks. It was estimated that each condition would take 30 minutes to complete. The experimental procedure for each test subject was as follows:

1. Test subject is given a 5 page technical guide demonstrating the use of the EvoBerry tool (See appendices A.84).
2. Test subject is given a demonstration of the key features of the tool by the experiment designer, using the tool.
3. Test subject is allowed up to 30 minutes to read through the technical guide, and to interact with the interface. The test subject is encouraged to perform mock searches with the tool, and investigate the different features.
4. The test subject is given an oral description of the experiment, as well as the tasks he will be required to perform. The test subject is told that the amount of monetary reward received is based on performance, and that there is no time limit. The test subject is told to inform the experiment designer if any queries arise during the experiment.
5. The test subject is assigned a scenario to perform using the EvoBerry tool. The user is told to inform the experiment designer after completing the questions to the best of his ability.
6. After completing the first scenario, the test subject fills in a questionnaire designed to gather qualitative information for the experiment (see appendices A.89 and A.90).
7. The test subject is given a 5 minute break.
8. The test subject performs the second scenario using the Internet Explorer web browser. The user is told to inform the experiment designer after completing the questions to the best of his ability.
9. Once the test subject has completed the scenario the experiment is over.

Search behaviour variables	
Variable	**Data recorded**
Searches:	Number of searches generated, search terms used in each search.
Search results:	Number of search results opened, rankings of each search result opened, URL address of each search result opened.
Web pages:	Number of web pages opened, URL address of each web page opened.
Tool usage variables	
Comparison tool	Number of clicks upon the tool.
Drop-box	Number of clicks upon the tool, number of objects stored in the drop-box.
Bookmarks	Number of objects bookmarked, URLs of objects bookmarked.
Session bar	Number of clicks upon the tool, number of sessions reopened, rankings of session bar objects.
Notepad	Number of clicks upon the tool.
Visual-history toolbar	Number of clicks upon the tool, number of bookmarks added.

Table 6.16: **Logging program variables.** These are the variables which were automatically recorded by EvoBerry's logging program. These included details of the users search behaviour (e.g. searches performed, web pages viewed) as well as users tool usage (e.g. number of uses of the comparison tool, drop-box). Note that the logging program was implemented after the first 5 experiments had been carried out. See appendix B.91 for more details.

(1) Tool usage questions
How often did you use the book marking function?
How often did you use the comparison tool?
How often did you use visual-history tool?
(2) Opinions on the different tools
Were the instructions in the walkthrough easy or hard to understand?
Was the comparison tool easy to use?
Was the visual-history function easy to use?
As a whole, was the tool easy or hard to use?
(3) Opinions on the interface in general
Was there anything that you didnt like or found confusing about the visualization?
Any other comments (e.g. any features you particularly liked?)

Table 6.17: **Questionnaire questions.** Three sets of questions were asked in the post-experiment questionnaire: (1) the amount of usage of certain tools in the interface, (2) the usability of certain tools in the interface (easy/hard), and (3) general opinions about the interface. The first two sets of questions consisted of line-length items, where users had to mark on a line to indicate their answer to the question. Text boxes were also provided for users to add comments. The final set of questions just used text boxes where users wrote their thoughts and opinions.

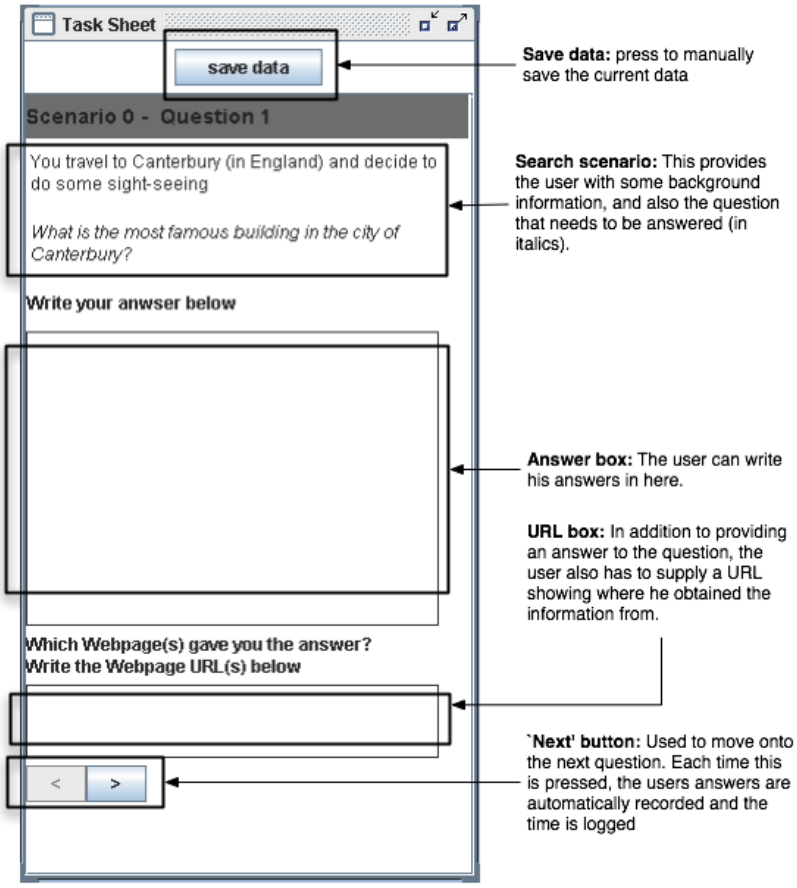

Figure 6.80: **Answer sheet program.** The same program is utilized in both the EvoBerry and IE conditions. This contains the user's search scenario and search tasks, which come in the form of questions that must be answered. Space is also provided for the test subject's answer. A separate text box is also provided for web page URLs. This ensures that test subjects use the tool to locate the information (as opposed to recalling the information from memory), forcing them to provide evidence of where they obtained the information from.

6.3.2 Experimental data

A substantial amount of data was gathered from the experiment. The primary concern of the experiment was to compare the performance of the two tools, which was done by recording the accuracy, time taken to complete the scenario, and number of searches generated by test subjects. In addition to this, information on other aspects of test subject's search behaviour (such as the number of search results opened, and web pages viewed), as well as the user's opinions on the tool, were also recorded and analyzed. Data was gathered by four different methods, (1) the 'answer sheet' program, which recorded the test subject's answers to the search tasks, (2) an inbuilt logging program that recorded all of the actions with the EvoBerry interface and its tools, (3) a screen capture program, that recorded the entire session as a video, and (4) a post-experiment questionnaire which gathered opinions on the tool, and any problems test subjects

experienced during their search.

Answer sheet

In both experimental conditions, an 'answer sheet' program was provided (written in Java). The answer sheet served a dual purpose: (1) it detailed the search scenario to users as well as the different search tasks that users had to complete, and (2) it provided space for users to input their answers to the search task questions. In the EvoBerry condition, the answer sheet was integrated onto the desktop of the EvoBerry tool as an internal frame, whereas in the Internet Explorer condition, the answer sheet is provided as a separate window. A screen shot of the answer sheet can be seen in Figure 6.80. When a user completed a search task (question) they click on the 'next' button to proceed to the next search scenario. Whenever the next button is used, the answers are automatically saved and the time is logged, so that the time taken to complete each question can be logged automatically.

Logging program

Having written the entire EvoBerry tool in Java, it was simple to generate a record of all of the user's interactions with the tool. In particular, the system automatically compiles a log of every instance a search was generated, which search results were looked at and what web pages were viewed by users. Tool usage data was also recorded, specifically the number of times each tool was clicked upon. A summary of the different variables recorded is summarized in Table 6.16.

Video capture

Each of the sessions were recorded using a screen capture program which recorded user's every movement on the screen. The screen captures were stored as a high-quality 640x480 AVI video. Two videos were recorded for each user (a separate video was made for each condition), with each video averaging around 600-900 megabytes. In total over 30 gigabytes of video data was obtained from all the experiments. The videos allowed the session to be replayed and analyzed in-depth, especially allowing post-analysis of how users utilized the tool and their search behaviour while interacting with the tool.

Questionnaire data

At the end of the experiment, test subjects were given an exit questionnaire. Three types of questions were posed on the test subjects tool usage, opinions of the different tools and their ease of use, and general opinions or comments about the interface. These questions can be seen in Table 6.17. A combination of line-length items and text boxes were used in the questionnaire. Test subjects marked their opinions using the line-length items, and provided more detailed

opinions in writing in the text boxes. When analyzing the results, their line-length items were divided into 3 equal sections, which were labeled positive, borderline, and negative, depending on the position marked. If no mark was made or it was stated that they never used the tool, then "never used" was marked.

6.4 Overview

This chapter has presented the design of an information-seeking experiment simulating an evolving search. The first half of the chapter establishes the distinction between an information retrieval and a information-seeking experiment, and the difficulties of designing an information-seeking experiment. The two challenges in designing an information-seeking experiment were (1) the creation of a 'difficult' search task, that could not be answered with the results of a single query, and (2) the problems brought about by the increased experimental time such as difficulties in acquiring test subjects, and difficulties in keeping test subjects motivated during the experiment. In the second half of the chapter, a detailed description of the experiment is presented.

Chapter 7

Results and analysis

This chapter presents the results of the experiment, and a detailed analysis of said results. The first half of the chapter consists of a presentation of the *experimental results*. An explanation of the different abbreviations used for each of the data variables is given and the data is presented in a series of tables summarizing the information in a readable format and observations on both the questionnaire and video data are given. The second half of the chapter consists of the analysis. The analysis is presented in three sections based on the different types of data: (1) *comparison data*, the data comparing the performance of the two interfaces, (2) *tool usage data*, the data comparing the usage of specific tools within each interface and (3) *search progress data*, which track changes in the user's performance through out the search session. The final part of the analysis takes a critical look at the *experimental factors* that could have influenced the results received.

7.1 Experimental results

A large amount of data was recorded during the experiment, for both EvoBerry and Internet Explorer interactions, and can be divided into five categories, *comparison data*, *tool usage data*, *search progress data*, *questionnaire data*, and *video data*. Each of these categories of data contains different variables that were recorded and analyzed.

Note that due to some technical issues a small amount of data (three out of forty videos) were corrupted and lost. While this does not affect the results which were recorded automatically (e.g. accuracy, time), other data (such as tool usage) could not be ascertained from these videos. See appendix B for full details.

Comparison data

The two interfaces are compared on the basis of the *time* taken to complete each scenario, the *accuracy* of the test subjects answers, the number of *searches performed* by the test subject, the number of *search results opened* (threads generated) by the subject, and *number of web pages* opened by the test subject. Table 7.18 shows a list of the variables and their respective abbreviations. All of the comparison data is reported in full in appendix B.91 and a summary of these results can be seen in Table 7.19, where the averages result for each variable is given.

The time in both conditions (EvT and IeT) is recorded in minutes. With regards to the accuracy variable, each scenario has 3 questions, the first and second question are both worth 3 points each because they ask users to find 3 pieces of information in each question. The last question is only worth 1 point because it only asks the subject to find 1 piece of information. In total each scenario is worth 7 points. Note that every time a user opens a search result from the search result list as a web page this is recorded as a search thread (as per the conventions laid down at the start of chapter 4).

	Abbreviations	
	EvoBerry	**Internet Explorer**
Time	EvT	IeT
Accuracy	EvA	IeA
Searches	EvSc	IeSc
Threads	EvTh	IeTh
Web pages	EvPg	IePg

Table 7.18: **Comparison data abbreviations.** Abbreviations of the comparison data variables recorded are provided above. The time in both conditions (EvT and IeT) is recorded in minutes. The accuracy of each test subject is a score between 0 and 7.

	time	accuracy	searches	threads	web pages
EvoBerry	36.35 mins	6.10	9.777	21.555	50.5
Internet Explorer	27.85 mins	6.05	11.684	17.263	54.789

Table 7.19: **Summary of the comparison results:** Above is a short summary of the results obtained from the experiment. The five main variables compared are time taken to complete the scenario (in minutes), the number of correct answers given (out of 7), the number of searches performed, the number of search results opened (number of threads followed), and the number of web pages viewed. Each variable is described in terms of its average score (mean). The entire data set and subsequent calculations can be found in appendix B.91.

Tool usage data

The data concerning the usage of EvoBerry functions was logged automatically each time a test subject used a specific function. The data on the use of the different Internet Explorer functions was recorded from observations of user's videos. A list of the abbreviations of the different functions whose usage was recorded is shown in Table 7.24. The tool-usage data is displayed in

	No. of uses (times clicked upon)				
	0	**1-5**	**6-10**	**11-20**	**21+**
History bar usage (no. of users)	15	4	0	0	1
Session bar usage (no. of users)	14	4	1	0	1
Bookmark usage (no. of users)	12	7	1	0	0
Dropbox usage (no. of users)	12	3	4	0	1
"Find" function usage (no. of users)	11	7	0	1	1
Comparison tool usage (no. of users)	8	9	1	1	1

Table 7.20: **Summary of EvoBerry function usage.** This table shows the number of uses (in clicks) of each tool by test subjects. The rows describe the different EvoBerry functions, along with how many of the 20 test subjects used the tool, and how many times they used each tool.

	No. of uses (times clicked upon)				
	0	**1-5**	**6-10**	**11-20**	**21+**
History usage (no. of users)	19	1	0	0	0
Bookmark usage (no. of users)	20	0	0	0	0
"Find" function usage (no. of users)	9	9	1	1	0

Table 7.21: **Internet Explorer function usage.** This table shows the number of uses (in clicks) of each tool by test subjects. The rows detail the different Internet Explorer functions along with how many test subjects used the tool, and how many times they used each tool.

	Abbreviations	
	EvoBerry	**Internet Explorer**
User progress - time	EvPT (1-3)	IePT (1-3)
User progress - accuracy	EvPA (1-3)	IePA (1-3)
User progress - searches	EvPSc (1-3)	IePSc (1-3)

Table 7.22: **Search progress data abbreviations.** The abbreviations of the search progress variables recorded are detailed above.

	time (means)			accuracy (means)			searches (means)		
	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
Evo	17.497	9.664	8.373	2.85	2.7	0.55	4.777	2.722	2.111
IE	14.519	6.814	5.96	2.7	2.75	0.6	5.473	3.105	3.368

Table 7.23: **Search progress data (means).** This table details the progress of test subjects when completing a scenario in terms of average time spent per question, average score per question and average searches generated per person. Results for both EvoBerry and Internet Explorer tools are provided. The entire data set and subsequent calculations can be found in appendice B.92 and B.93.

full in appendix B.94, and a summary of this data can be seen in tables 7.20 and 7.21. Each tool's usage data is reported in terms of the number of times test subjects interacted with the data, recorded by the computer as the number of times they clicked on that particular function. The usage is grouped into several blocks (0, 1-5, 6-10, 11-20, and 21+ uses).

	Abbreviations	
	EvoBerry	**Internet Explorer**
Browser history	EvHs	IeHs
Bookmark	EvBm	IeBm
Find function	EvFd	IeFd
Session bar	EvSb	N/A
Dropbox	EvDb	N/A
Comparison tool	EvCo	N/A

Table 7.24: **Tool usage data abbreviations.** The abbreviations of the tool-usage variables recorded are detailed above. Note that only the first 3 tools have comparable counter parts in both interfaces. In the row 'browser history' the visual-history toolbar is compared with Internet Explorer's browser history tool. EvBm specifically refers to the combined usage of the results frame highlighting bookmark, and the visual-history's bookmark function.

Search progress data

Information detailing test subject's progress at different points in the search was also recorded. This includes the time taken to complete each question in the scenario, accuracy on a question by question basis, and the number of searches needed to answer each question. These are useful for following test subjects progress over time with regards to a specific variable. Note that each variable abbreviation is followed by a number between 1 and 3, which is used to identify the question that the variable refers to e.g. EvPT1 refers to the time taken by test subjects to complete question 1 in the EvoBerry scenario. This data is reported in full in appendix B.92 and B.93. A list of the abbreviations used for these variables is listed in Table 7.22 and a summary of the results can be seen in Table 7.23.

	Tool frequency of use			
	frequently	moderately	rarely	never used
Comparison tool	1	1	7	11
History-bar	6	2	6	6
Bookmark (highlighting)	2	3	7	8

Table 7.25: **Questionnaire results (frequency of use):** This information was extracted from one of the sections of the post-experiment questionnaires. This particular section refers to the frequency of use of specific tools by test subjects.

7.1.1 Questionnaire data

The questionnaire data was used to capture test subject opinions on the performance and appearance of the Evoberry interface and its tools. The quantitative data is summarized in Tables

	Tool ease of use			
	Positive	Borderline	Negative	Never Used
Comparison Tool opinion	13	3	0	4
History bar opinion	12	3	1	4
Walkthrough	18	1	1	n/a
Entire Tool opinion	16	2	2	n/a

Table 7.26: **Questionnaire results (ease of use):** This information was extracted from one of the sections of the post-experiment questionnaires. This particular section refers to the ease of use of specific tools, as reported by test subjects .

7.25 and 7.26. Through the questionnaires, test subjects provided interesting and varied feedback, as well as help in identifying benefits and flaws in the design. Note that throughout the rest of this thesis, references to individual test subjects will be made using their code numbers (A01 to 20). Below is a summary of the main comments reported by the subjects.

- **Comparison tool:** Test subjects reported positively about the comparison tool[1]. Of all the EvoBerry tools, it was the most used, and was the most highly praised, and only positive comments were received about the tool. Subject A13 commented positively on how his searching patterns were changed by the comparison tool, enabling him to drill down and discover pertinent results more quickly.

- **Web Tab Panel:** The fixed space that comprised of the web tab panel came under a large amount of criticism[2] due to its fixed position (although it can be resized) which meant that all other windows in the workspace had to be reorganized around it. The fact that all the web pages are displayed in this space, and that test subjects spend a large portion of their time viewing web pages, meant that the size of the panel had to be constantly resized back and forth. The sheer amount of room taken up by the web tab panel (combined with its fixed position) meant that results frames often became cluttered in the left hand side of the workspace. One complaint (by subjects A02, A04, and A07) was that, if one was not careful, results frames would sometimes be generated behind the web tab panel because of the fact that a splitpane method was used (see the design in chapter 5 - subsection 5.2.2 for the reasons behind using the splitpane).

- **Results frames:** There were mixed comments about the results frames. Some test subjects enjoyed (subjects A03, A17) the element of organization introduced by creating a division between the search results and web pages. Others thought that because of their size and the lack of space brought on by a fixed web panel (see the previous comments on

[1]Subjects which reported positively about the comparison tool were, A01, A04, A06, A07, A10, A13, A14, A19, and A20

[2]Subjects which reported negatively about the web tab panel were, A07, A11, A17, A18, A20

web tab panel), the workspace became cluttered far too quickly. The coordinated colouring between the results frames and their spawned web pages in the web tab panel was commended by a couple of subjects (subjects A03, A19), stating that it aided them in re-identifying their tabs correctly and keeping track of their searches.

- **History tools:** Both tools (the visual-history toolbar and the session bar) were under-utilized. The visual-history tool was praised (by subjects A11, A12) because it did not lose any web pages generated by forward movements, if users moved backwards. The session bar received some criticism (from subjects A4, A14) over the size and complexity of its icons. Unlike many of the other controls (such as those for the results frame), these were designed to be small and inconspicuous, because the session bar had to be capable of displaying a large number of objects visibly.

- **Other tools:** No comments were made about the other Toolbox tools (note pad and the drop box), which is not surprising given the secondary roles of the tools, as short term data storage containers. Observations of search behaviours showed that the lack of usage of the notepad was due to the fact that the answer sheet was often used instead to write notes on. Another tool which was mentioned was the "Find" tool. Although a side feature of the tool, it was nevertheless used extensively by test subjects in both conditions. A couple of subjects (A14, A15) complained that the find tool should be more instantly obvious (CTRL-F had to be pressed to summon it).

- **Search Behaviour:** A comment made by test subjects (A07, A10, A13), was that they felt they were under-utilizing the tools provided. This much is evident from the results on tool usage presented earlier in this section (see Table 7.20). It was also noted that time and continued usage was needed in order to become more familiar with the tools, and be able to use them more efficiently.

- **Experimental design:** One comment (by subject A13) was made about the pre-experiment training; it was felt that more time could have been allocated to the training, and perhaps a mock-search could have been provided for test subjects to attempt to complete. There was some confusion over the role of the answer-sheet window: some test subjects assumed that it was a natural part of the program, and were confused when they discovered no shortcuts had been provided for sending information to the answer sheet.

- **Technical errors:** A small number of technical errors were reported during the sessions (such as crashes, and tools doing unexpected things). All technical errors were investigated throughout the course of the experiment and were isolated and removed. The most common error was that the interface crashed unexpectedly, a fault later linked to a virus

on the computer which was affecting Internet Explorer's rendering of the web page in the web tab panel. This was subsequently remedied. Some technical errors occurred from test subjects performing unexpected actions, such as submitting a second query before the first query had resolved. Simple measures could be taken (such as locking down the input system while a search was being performed) were introduced to counter these effects.

The single largest criticism[3] with the interface was the split-pane in which the web tab panel resided. Test subjects commented that they grew frustrated with having to resize the split-pane when switching between the results frames and the web pages. They expressed the opinion that a free-floating window that could be moved around and overlaid (like the other internal frames on the desktop) would be more beneficial. Another suggestion was the use of fixed components in a fixed layout that did not need to be moved around and a suggestion was made for tabbing the results frames, so that they could be organized easier.

Reports during the sessions indicated a decline in EvoBerry's speeds during the course of the search. It was noted, especially towards the end of their sessions that, as the number of searches performed increased, the speed of the tool decreased. This is attributed to poor internal tool design, while the speed of certain facets of the tool (such as the data-gathering) were looked at thoroughly, others (such as data handling over time) was not looked at so closely. The preliminary test runs with the tool did not flag this problem since (1) they mostly looked at surface problems with the interface and (2) evaluators of the tool in the preliminary tests neither spent as much time using the tool, nor created as many searches as the test subjects in the experiment. The exact cause of the slow down is as yet undetermined, although it has been suggested that the problem is associated with a lack of scalability in the internal data structure. These speed issues will be looked at more closely in future versions of the tool.

7.1.2 Video data

A large amount of video data was gathered from the search sessions and was useful in observing the search behaviour's of the users. Review of the videos called attention to several different recurring behaviours, present in the search session's of users. These behaviours were related to (1) user interactions while generating searches, (2) user interactions with web pages, (3) user interactions with the comparison tool, and (4) user interactions with the workspace.

Behaviour 1 : search generation

Two distinct behaviours were noted when observing users generating searches. In the first behaviour, a large number of searches are generated very quickly. and after the search results

[3]Subjects which complained about usage of the split-pane were, A07, A11, A17, A18, and A20

are returned, the first ten results will be quickly skimmed through. If pertinent information is spotted, then web pages will be opened and viewed, otherwise the query will be reformulated immediately and a new set of search results generated. The behaviour was named *impulsive search behaviour*, because of the speed at which users would generate and discard searches without thoroughly inspecting the returned search results. The behaviour was noted in a number of users in the Internet Explorer condition[4].

The second search behaviour only applied to a small number of users (test subjects A04 and A15), where users displayed a search behaviour which ran contrary to those displayed in an *impulsive search behaviour*. In this case the user would generate a search and then spend time opening a number of search results and investigating them thoroughly. This was named the *exploratory search behaviour* and is characterized by generating few searches, but exploring each of these lists of search results thoroughly.

Behaviour 2 : web pages

The video data also highlighted a pattern specifically related to the way users view web pages. A small number of users (subjects A06, A13 and A20) followed a consistent pattern whereby they would perform searches, and open search results (generate threads) but for the most part only view one or two web pages (if any) from that search result before returning to the main search result list. Once or twice during their search, the users would concentrate on a single thread, opening between 10 and 20 web pages from a single search result. It would appear that these users perform only a cursory inspection of the majority of search results that they open (visiting only one or two web pages), but once they find a search thread that interests them they will investigate it thoroughly, opening a large number of web pages. This pattern of search behaviour bears much resemblance to the *depth-first* search pattern from Tauscher and Greenburg's [114] search patterns (as seen in chapter 2 - subsection 2.2.3).

Behaviour 3 : comparison tool

Analysis of the videos showed that some test subjects (subjects A06, A13) relied solely on the comparison tool; generating searches and consulting the comparison tool (to find any search intersections) before actually viewing the results in the search results list. However one consequence of this search behaviour was that test subjects often performed a large number of searches to create these intersections, which in turn cluttered the screen with results frames. It was interesting to note that not only were many of the users experimenting with the tool, but were also applying it to helping solve their search scenarios.

[4]Subjects displaying *impulsive search behaviour* in the Internet Explorer condition were, A01, A02, A06, A07, A10, A12, A13, A14, A15, A16, A18, A19, A20

Behaviour 4 : workspace

The video data also provided an opportunity to observe the behaviours of users when managing their data. When observing test subjects using the EvoBerry tool, two contrary behaviours were identified. A small set of test subjects were noted for their strict management of windows and web pages, only keeping active results frames and web tabs open and in clear view, while closing all unused frames and tabs. Running in contrast was another group, which showed little or no management of their information, this group would generate many results frames and web pages, leaving them open overlaying each other and cluttering up the view of the workspace.

When observing test subjects using Internet Explorer, another distinct search behaviour was noted. A small group of users (test subjects A02, A03, A04, A15, A17 and A18) would utilize either two or three explorer windows and continually switch usage between these different windows. Often these windows were seen to be used in parallel, often for comparing information from two different sources. The best example of this performed by multiple users in the thailand holiday scenario, was using one window to open search results from the search engine, while another window was used to display a map or list of airports.

At this stage only a cursory examination of the search behaviour's of users was possible due to time constraints and a greater emphasis being placed on analyzing the comparison and tool usage data. Future analyses will focus more upon these qualitative aspects of the search behaviour of users. These behaviours will be discussed again later in the chapter with regards to the to the comparison and tool usage data. The behaviours observed in the videos combined with the statistical data from the logs, help to explain the performance differences between the two conditions (EvoBerry and Internet Explorer).

7.1.3 Overview

The results received were positive, but complex in nature (see Table 7.19). It is a clear that not all of the hypotheses were fulfilled, but the results did not deviate far from the expected outcome, and in some aspects, the tool exceeded the expectations. A preliminary examination of the results, in comparison with the experiment's hypotheses, indicates the following:

1. Test subjects (on average) completed the search scenarios faster in the Internet Explorer condition than in the EvoBerry condition.
2. Test subjects (on average) were equally accurate in both the Internet Explorer and EvoBerry conditions.
3. Test subjects (on average) performed less searches in the EvoBerry condition than in the Internet Explorer condition.

The fact that test subjects were faster in completing the search scenario in the Internet Explorer condition was not surprising. Much can be explained by the difficulties of comparing experimental interfaces, given the differences in tool usage experience. More surprising was the fact that, in spite of test subjects relative inexperience with the tool, they still achieved scores through EvoBerry that equalled (and sometimes surpassed) their scores in the Internet Explorer condition.

However this simple analysis still leaves many questions unanswered, for example, why do test subjects create less searches when using EvoBerry than when using Internet Explorer, but open more search results? Was test subject inexperience the only factor affecting the speed, or did particular tools in the interface aid or hinder search progress? An in-depth investigation of the results is necessary. The data presented is further analyzed using the following techniques:

Descriptives

The data descriptives provide a simple and quick method of analyzing the data. In addition to the previously used mean (average), a number of other data descriptives such as the median, mode, standard deviation and range are presented in this analysis of the data.

T-test analysis

Solely calculating the means is not an effective method to describe the difference between two data sets, because it does not provide a measure of what constitutes a 'significant' difference. One method (mentioned already in chapter 4 - subsection 4.2.3) is Welch's t-test [26], which can be applied to two data sets to determine the significance of the difference of the results. Note that Welch's t-test was chosen over the more commonly used 'Student's' t-test because an equal variance between the two groups is not assumed. A two-tailed paired t-test was used, with an alpha value of 0.05. A two-tailed test was chosen because there was equal chance of the parameters shifting in the opposite direction of the hypotheses (e.g. there was an equal chance of EvT > IeT and IeT > EvT). A paired test was performed since the results being compared both originated from the same source (test subject). The alpha-value chosen is standard measurement in most t-tests.

What is important to remember, is that the result of a t-test is the *p-value*, a measure of the significance of the difference in the results. The p-value represents the percentage chance that the results of the experiment would be different if the exact same experiment was performed on a completely different set of test subjects, e.g. a p-value of 0.01 signifies that there is a 1% chance of a different outcome. As a result, the lower the p-value, the better. As a general measure, the p-value is often compared against the alpha value (in this case 0.05), and a p-value equal or lower than the alpha-value is considered a significant difference.

One of the problems with any form of analysis is that the accuracy and significance of the results is very much dependent on the number of results available, and in general the higher the number of results being compared the more significant the difference in the results. In the experimental design (chapter 6 - subsection 6.3.1) it was commented that the number of subjects used in the experiment was not ideal, and that twice that number (40 or 50 test subjects) would yield more significant results.

Correlation

A correlation between two variables helps identify relationships between different variables, and in turn can help explain performance differences in the comparison data, as well as link the usage of specific tools to changes in performance. A method of finding the degree of correlation between two sets of data is the *Pearson product-moment correlation coefficient* [26] (also known as PMCC). The PMCC is usually denoted by the value r.

The value r is a measure of the extent to which two measurement variables "vary together", and must be a value between -1 and +1. The magnitude of r "indicates" the strength of the relationship, and the sign of r reveals the direction of the relationship. A value of -1 indicates a perfect negative relationship where all data points lie on a single line but that Y increases as X decreases. A value of +1 indicates a perfect positive relationship, where all data points lying on the same line and with Y increasing with X. An r of 0 shows that a linear model is inappropriate that there is no linear relationship between the variables. For the purposes of this analysis, an r of value 0.80 (positive or negative) or higher will be of sufficient strength to be considered a correlation (based on [66]). An r value of greater than 0.5 will be considered a much weaker correlation, but a correlation none the less.

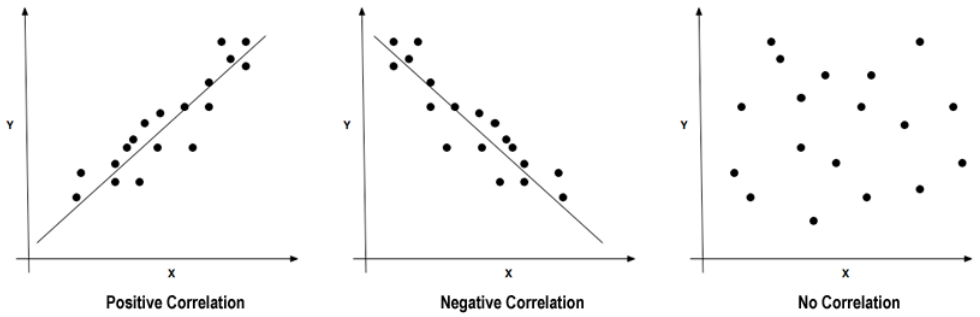

Figure 7.81: **Types of correlation.** On the left is a positive linear correlation (as x increases so does y), in the centre is a negative linear correlation (as x increases, y decreases) and on the right is a line with no correlation.

Using these three techniques, analysis of the three sets of data (comparison, tool usage, search progress) was performed. The following three sections will look at each of these sets in

detail, applying the techniques and discussing the results.

7.2 Comparison data

The initial analysis consisted of comparing seven descriptives of the data, which either give the data a representative value (mean, median, mode) or measured the size and variability of the data (standard deviation, range, minimum, maximum). The comparison data descriptives are provided below.

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvT	36.35	34	33	8.677	35	17	52
IeT	27.85	23	19	12.368	47	13	60

Table 7.27: **Data descriptives: time**

Table 7.27 compares the descriptives of EvT and IeT. There is a large difference in the average time, as shown by differences in the mean, median and mode, with test subjects in the Internet Explorer condition completing scenarios quicker. The descriptives also show that the IeT variable has both a larger range and standard deviation than EvT. While on average, test subjects performed faster in the IeT condition, their results showed less consistency and greater deal of variability.

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvA	6.1	6	6	0.852	3	4	7
IeA	6.05	6	7	1.145	4	3	7

Table 7.28: **Data descriptives: accuracy**

Table 7.28 compares the descriptives of EvA and IeA. There is a great deal of similarity between the two sets of results (as demonstrated by similar means and medians), which is not surprising given the extremely small ranges of the data. The significance of the difference between these two sets of data cannot be established easily using these descriptives. When compared with the differences in time (as demonstrated by EvT and IeT), it can be seen that test subjects in the Evoberry condition took longer, but performed equally well. As mentioned previously, this can be seen as a positive result, showing that test subjects perform equally well using an unfamiliar tool. However it is still unclear as to whether the time difference was a result of the users unfamiliarity with the tool, or some other factor e.g. a tool, or search behaviour specific to the EvoBerry condition.

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvSc	9.777	10	7	3.888	12	4	16
IeSc	11.684	9	9	6.591	23	4	27

Table 7.29: **Data descriptives: searches**

Table 7.29 compares the descriptives of EvSc and IeSc. The descriptives show that on average test subjects in the Evoberry condition generated less searches than in the Internet Explorer condition. However, the difference between the means is less than 2 searches, and it is difficult to determine whether this is indeed a significant difference. The range of results seems to also be much larger in the Internet Explorer condition, than in the EvoBerry condition. Note that while EvT is greater than IeT, EvSc is actually lower than IeSc. In essence, test subjects in the EvoBerry condition spent a longer time on the search scenario, but performed less searches, which goes against the conventional wisdom that creating more searches, will increase the amount of information users have to view, and furthermore increase the amount of time taken to complete the scenario.

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvTh	21.555	17.5	25	11.627	47	9	56
IeTh	17.263	14	12	8.830	36	9	45

Table 7.30: **Data descriptives: threads**

Table 7.30 compares the descriptives of EvTh and IeTh. The results show that on average test subjects in the Evoberry condition created more threads than in the Internet Explorer condition. Both the median and mode seem to suggest that test subjects create a larger number of threads in the EvoBerry condition. Although EvTh and IeTh show similar minimums, EvTh has a far larger range. Note that both EvT and EvTh are greater than IeT and IeTh respectively. It is possible that the increase in EvT is not related to the number of searches that test subjects create, but is instead a product of the number of search results opened (and as a result, number of search threads created). This might also suggest that test subjects adopt different search behaviours while using the different interfaces, EvoBerry users generating less searches than Internet Explorer users, but viewing more of the search results.

Table 7.31 compares the descriptives of EvPg and IePg. The results show that on average test subjects open more web pages in the Internet Explorer condition. An extremely large amount of variability in the data was reported, with a range of 160 for IePg (compared to the 90 in EvPg), and while both sets share similar minimums, the difference in maximums is great.

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvPg	50.5	46	65	24.103	90	23	113
IePg	54.789	41	-*	38.980	160	19	179

Table 7.31: **Data descriptives: web pages:** -* note that the mode of IePg is not available, since no single page count appears more than once in this data set.

Given such a large difference in ranges, it is difficult to say whether the means are an effective representation of the data. Note that the web page data shares some similarities with the search data, where EvSc > IeSc and EvPg > IePg.

7.2.1 Analysis : comparison data

In this analysis the following assumption is made; during a search session, a test subject's time, searches, threads and web pages will all share a positive linear relationship, i.e. as they create more searches, the time, threads opened and web pages generated will also increase. In the descriptives, different trends were identified, some of which conformed to this assumption and others which ran contrary to this assumption.

The trends which conformed to this assumption were the relationships between time and threads (EvT > IeT and EvTh > IeTh), and between searches and web pages (EvSc > IeSc and EvPg > IePg). Given the evidence from the first trend, it was postulated that a relationship between time and threads exists in both conditions, thus whatever factor caused the differences in experimental time also created a similar difference in the amount of threads generated (and vice versa). If such a relationship is assumed to exist, then differences in test subject's performances can more easily be explained, e.g. the reason test subjects spent more time completing scenarios in the EvoBerry condition may be related to whatever factor caused them to view more search results. A similar theory could be postulated for the relationship between test subject's searches and web pages. This theory is explored further in this section.

Several trends ran contrary to the stated assumption. These were the relationships between time and searches (EvT > IeT and EvSc < IeSc), threads and searches (EvTh > IeTh and EvSc < IeSc), time and web pages (EvT > IeT and EvPg < IePg), and threads and web pages (EvTh > IeTh and EvPg < IePg). It was hypothesized that the first two trends (time and searches, and searches and threads) are linked. The relationship between time and searches shows that although test subjects spent a longer time using the EvoBerry tool (than with Internet Explorer), they created less searches with the EvoBerry tool. This might suggest that when utilizing EvoBerry, test subjects (on average) spend more time browsing the results of their searches, than when utilizing Internet Explorer. The relationship between searches and threads shows that when utilizing the EvoBerry tool, test subjects created less searches, but

opened up more search threads (viewed more search results). This seems to support the previous implication, that test subjects are spending more time exploring their data (and creating less searches) than when using Internet Explorer, in particular spending more time viewing and opening search results.

These two trends bear a resemblance to the afore mentioned *impulsive* and *exploratory* search behaviours (as detailed in the video data - section 7.1.2). The progress of the test subjects in the EvoBerry condition is characteristic of an exploratory search behaviour, resulting a large number of search results and threads being generated. The progress of test subjects in the Internet Explorer condition is characteristic of an impulsive search behaviour, which results in more searches, but fewer threads. The existence of impulsive search behaviour in the Internet Explorer condition is further supported by the fact that even though IeSc is higher than EvSc, EvT is significantly higher than IeT, a trend that would appear if test subjects were 'impulsively' generating searches, and very quickly discarding their results and reformulating. Part of the cause of this shift in search behaviour may be a result of the difference in the technologies of the two interfaces. On the basis that the Internet Explorer tool is quicker at performing searches than the EvoBerry tool (by only 5-10 seconds or so), it is possible that the interface encouraged test subjects to reformulate searches more often, given the little effort required, and in turn forming a more impulsive search behaviour.

The last two trends (time and web pages, and threads and web pages) also appear to be linked. The relationship between time and web pages shows that although test subjects spent a longer time using the EvoBerry tool, they viewed less web pages than in Internet Explorer. The relationship between threads and web pages shows that although test subjects using EvoBerry opened more search results, they viewed less web pages. A theory could be posited to explain these two trends, based on two different search behaviours in Tauscher and Greenburg's [114] search patterns (as seen in chapter 2 - subsection 2.2.3), *depth-first* and *breadth-first* searching.

When utilizing Internet Explorer, a depth-first search is adopted, where few search results are opened, web pages from these search results are explored in depth, and many links from these web pages are followed onto other pages. When utilizing EvoBerry, a breadth-first search is adopted, where many search results are viewed and opened, but not explored in detail, instead concentrating on opening more search results. The two sets of behaviours (impulsive/exploratory search behaviours and depth-first/breadth-first searching) exhibit similarities, but the relationship between the two sets of behaviours is not all-inclusive, e.g. test subjects may exhibit the characteristics of an impulsive search behaviour, but not depth-first searching. Video evidence (see section 7.1.2) was discovered confirming the existence of these behaviours in small groups of test subjects.

The data descriptives allow quick conclusions to be formed, without having to inspect the

results in-depth, an advantage when dealing with large amounts of data. However there are limitations to an analysis based only on the descriptives of the data, because the results can sometimes be misleading. For example, it is possible to skew the mean of a set of results by including a significantly large or small data point (which would normally be considered an outlier). This limitation in the means, can in turn affect assumptions about (1) relationships established between sets of results, and (2) the significance of differences between sets of results. The following two subsections will apply Pearson's correlation and Welch's t-test to the comparison data.

variables	***r***
EvT - IeT	-0.012
EvA - IeA	-0.143
EvSc - IeSc	0.037
EvTh - IeTh	-0.180
EvPg - IePg	0.384

Table 7.32: **Pearson's correlation : between conditions.** This shows the correlations between variables in the two different conditions. Note that the closer a correlation is to 1.0 or -1.0, the greater the correlation. Correlations close to 0.0 mean there is no relation between the two variables.

EvoBerry variables	***r***		*Internet Explorer variables*	***r***
EvT - EvA	0.112		IeT - IeA	0.005
EvT - EvSc	0.173		IeT - IeSc	0.274
EvT - EvTh	0.483		IeT - IeTh	0.658
EvT - EvPg	0.451		IeT - IePg	0.688
EvA - EvSc	-0.064		IeA - IeSc	0.119
EvA - EvTh	0.003		IeA - IeTh	0.377
EvA - EvPg	0.043		IeA - IePg	0.094
EvSc - EvTh	0.182		IeSc - IeTh	0.596
EvSc - EvPg	-0.084		IeSc - IePg	0.478
EvTh - EvPg	0.301		IeTh - IePg	0.641

Table 7.33: **Pearson's correlation: within conditions.** On the left shows the correlations of the EvoBerry condition variables, the right shows the correlations of the Internet Explorer condition variables.

Correlation analysis

Pearson's correlation[5] was applied to variables *between conditions* (Table 7.32), and variables *within conditions* (Table 7.33). Between conditions looks at the correlations between two variables in different conditions, e.g. correlating time taken between the EvoBerry and Internet Explorer conditions (EvT and IeT). Within conditions looks at correlations between different variables when using the same tool, e.g. correlating time taken with searches performed for users of the EvoBerry tool (EvT and EvSc). Looking at the results of correlation (both between and within conditions) no strong correlations can be found (r ¿= 0.80), however a number of

weaker correlations were noted (IeT - IeTh, IeT - IePg, IeSc - IePg, and IeTh - IePg).

At the start of the chapter it was posited that a relationship existed between time and threads in both conditions, on the basis that EvT > IeT and EvTh > IeTh. A theory could be posited to explain the correlation between time and threads, based on the different interfaces used by EvoBerry and Internet Explorer. The EvoBerry interface displays fifty results at a time, while the Internet Explorer interface only displays 10 results at a time, meaning there is a difference in the number of search results that can be opportunistically viewed. By increasing the number of search results in the view, the EvoBerry interface is encouraging test subjects to spend more time viewing and opening search results, thus increasing the time spent in the EvoBerry condition.

Looking at Table 7.33, correlations to support this theory can be seen in both the Internet Explorer (IeT - IeTh) and EvoBerry (EvT - EvTh) conditions, although these are both weak correlations and thus the theory is difficult to prove and will require further experimentation and more experimental data. It was also posited that a relationship existed between searches and web pages in both conditions, however a correlation was only found in the Internet Explorer condition.

compared variables	**p-value**
EvT - IeT	0.007
EvA - IeA	0.880
EvSc - IeSc	0.422
EvTh - IeTh	0.203
EvPg - IePg	0.911

Table 7.34: **T-test results of the comparison data:** A two-tailed, paired t-test was performed on the comparison data to determine the significance of the differences in the results.

T-test analysis

A two-tailed paired t-test analysis was performed on the comparison data, the results of which can be seen in Table 7.34. Of the t-test values only the difference between the times of the two sets of data was seen to be highly significant (p-value = 0.007). At the opposite end of the scale, two of the sets of results have p-values very closely approaching 1.0, the accuracy data (p-value = 0.88) and the page data (p-value = 0.91). While this does not mean that the results are significantly similar, it does show that the means for the accuracy and web pages are very much indistinct, and that the two groups do not differ by a significant margin. These results have implications with regards to previous theories.

[5]Before Pearson's correlation was applied, each set of data was screened for outliers, to increase the accuracy of any correlations found.

Previously it was posited that a relationship existed between the time taken and threads generated. A theory based on the assumption that both, times (EvT and IeT) and threads generated (EvTh and IeTh) were significantly different. There is a significant difference in time (p-value = 0.007) but the difference between the threads is not close enough to significance (p-value = 0.203) to guarantee that this theory is valid.

Two theories were also posited which stated that the search behaviour (and hence their search data) of test subjects differ based on the interface being utilized. The theory of impulsive/exploratory search behaviour was based on differences in time, searches, and threads. The completion times in each condition (EvT and IeT) were significantly different, however both the differences in number of threads (EvTh and IeTh) and number of searches (EvSc and IeSc) generated in each condition were not significant enough to validate this theory (p-value's of 0.203 and 0.422 respectively). These results only partially support the theory, and it is unclear as to whether an increase in the number of experimental results will bring these differences closer to significance.

The theory of depth/breadth-first search behaviours was based on differences in time, threads and web pages. The completion times in each condition (EvT and IeT) were significantly different, however the differences between the number of threads (EvTh and IeTh) generated and web pages viewed (EvPg and IePg) were not significant. Given that two of the results (threads and web pages) are not significant, it is unlikely that enough of a difference exists to confirm this theory.

Overview: comparison data

An analysis of the comparison data was performed, looking at the descriptives of the data, the correlations between different variables, and the significance of the differences between the data sets. The analysis identified two areas of interest, (1) the relationship between time and threads, and (2) the relationship between searches, threads and web pages. It was theorized that the difference in time (between EvT and IeT) was a result of whatever factor caused a difference in the threads generated. This was of particular interest because it would help explain why the results (EvT > IeT) differed from the original hypothesis (EvT < IeT).

It was posited that thread generation was linked to differences between the two interfaces, specifically the number of search results displayed. EvoBerry displayed a larger amount of results, and thus encouraged test subjects to spend more time exploring search results. The evidence only partially supported this theory, the t-test showed a significant difference in only one of two the pertinent variables (time), and although correlation existed in both variables, it was a weak correlation.

A trend was exhibited related to the differences in time, searches, threads and web pages,

which ran contrary to previous assumptions. Two theories were presented to explain these differences, relating to impulsive/exploratory search behaviour, and depth-first/breadth-first search behaviour. A review of the video evidence shows the existence of these behaviours among small groups of users, however these small numbers combined with a low level of significance in the difference between the groups make it difficult to prove these theories.

The analysis has galvanized a number of interesting theories on the reasons for the differences in the comparison data. However, evidence to support these theories is still weak, and more experimentation needs to be carried out to gain more statistically significant results. It is possible that the performance of the users may have been affected by other experimental and technological factors. The analysis is continued in the following sections, looking at the tool usage and search progress data each in turn.

7.3 Tool usage data

The data showed that the tools of both interfaces were underutilized. Table 7.21 shows that with the exception of a single tool (the 'find' function), the majority of test subjects did not utilize the tools provided in Internet Explorer. Similarly, looking at Table 7.20, it can be seen that with the exception of the comparison tool, all of the EvoBerry tools were only utilized by 25-50% of the test subjects. These results are not surprising. It has been established in browser history system literature (see Tauscher [114], Berkun [11], and Kaasten [60]) that web browsers history tools (such as the bookmark tool, and history) are under-utilized. The research seems to indicate that bookmark system are more often used for long-term recall, where users knows they will revisit the web page many times. In the case of short-term recall, the research (as well as our own observations) show that users will more often attempt to recreate previous searches when recalling previous information, rather than utilize the browsers history tool.

Previous research (see [110]) has shown that it is difficult to encourage test subjects to use unfamiliar tools (such as the Evoberry tool). A comment that showed up in the questionnaire results, (subjects A04, A10), suggested that some found it difficult to adjust to the tool, and confessed that once they were focused on the task, they forgot about the EvoBerry's tools and used the interface as if it was Internet Explorer. However this opinion was not shared by all. A large number of test subjects (as illustrated by the questionnaire results in section 7.3.1) expressed interest in the comparison tool, citing it as a useful tool. The results of the tool usage data (see Table 7.20) show that the comparison tool was the most utilized of all the EvoBerry tools.

	Comparison data variables				
Tool usage	**Time**	**Accuracy**	**Searches**	**Threads**	**Web pages**
Session bar	0.235	-0.495	-0.197	-0.065	0.101
Visual-history	0.483	0.317	-0.061	0.033	0.638
Bookmark	0.271	-0.194	0.072	-0.018	-0.232
Drop box	0.038	0.048	-0.083	0.350	0.275
Comparison tool	-0.195	0.412	-0.108	-0.278	-0.034
Find tool	-0.448	0.139	-0.216	0.011	-0.027

Table 7.35: **Correlation of comparison and tool usage data.**

7.3.1 Analysis : tool usage data

Pearson's correlation coefficient[6] was applied to the EvoBerry tool usage data in order to determine any relationships between tool usage and user performance (shown in table 7.35). The data showed no strong correlations between user performance and tool usage, however a number of weaker correlations were identified. A weak positive correlation was identified between visual history usage and web page generation (EvPg - EvHs). This correlation implies that as the visual-history usage increases, so does the number of web pages viewed, but it is difficult to determine whether the increased web page views was a positive effect (the tool encouraged more detailed browsing of web pages) or a negative effect (the tool hindered test subjects efforts forcing them to open more web pages).

It is interesting to note that several observations made in both the questionnaire and video data relate to some of the weak correlations identified in table 7.35. While these correlations have r values below the guideline set earlier in the chapter (an r of at least 5.0), they still provide an interesting outlook on how tool usage may affect the users performance. Two such correlations linked increased tool usage to increased user performance. A weak positive correlation was identified between usage of the comparison tool and user accuracy, implying that comparison tool usage increased user accuracy. This may help to explain why the tool has been praised by the majority of its users (see questionnaire data - section), and had the highest usage of all the EvoBerry tools (see the tool usage descriptives - subsection). A weak negative correlation was identified between usage of the find tool, and time taken by users, implying that increased find tool usage lowers the time taken for users to complete scenarios. It was posited that using the find tool allowed test subjects to cut down on the time taken to view web pages, and as a result lower the time taken to complete scenarios. Interest was garnered in the find tool because of its high usage by test subjects, and observations during the experiments, as well as questionnaire and video data showed that not all users were reminded of the functions existence because it's existence is not explicit (there is no graphical widget or button representing the tool on the desktop) and test subjects must remember to use Ctrl-F key combination to activate it when in the tabbed browser.

There was also a weak negative correlation between the accuracy of test subjects and session bar usage. This correlation implies that as session bar usage increases, accuracy decreases. This is supported by questionnaire data showed that test subjects exhibited a certain amount of dissatisfaction with the session bar tool, with criticism of its visibility and complexity.

Overview

Analysis of the tool-usage data has identified interesting trends related to four of the tools present in the Evoberry interface and has highlighted the fact that different tools in the EvoBerry interface have different effects upon search progress. Two of the tools were identified as having a positive influence on search progress, namely the comparison tool, and the find tool. Two of the tools were identified as having a negative influence on search progress, namely the session bar and the visual-history. At the moment the correlations identified have been weak, but are supported by observations in both the questionnaire and video data.

7.4 Search progress data

The search progress data allowed the analysis of changes in the test subject's accuracy, time and searches over the course of the search session, in both conditions. This information is useful when attempting to isolate behaviours or trends which are specific to certain parts of the search session. The search progress data records test subject's accuracy for each search task, time spent on each task, and the number of searches generated during each task in the search scenario. The initial analysis of the search progress data consisted of looking at the data descriptives.

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvT1	17.497	15.68	13.57	6.882	31.34	5.36	36.7
IeT1	14.519	12.56	N/A	9.103	31.22	2.3	33.52
EvT2	9.664	8.955	N/A	4.180	15.8	0.1	15.9
IeT2	6.814	5.2	2.25	4.747	18.9	2.25	21.15
EvT3	8.373	8.365	N/A	3.585	12.86	2.92	15.78
IeT3	5.96	4.21	N/A	5.687	20.93	1.15	22.08

Table 7.36: **Data descriptives: EvPT - IePT.** This set of data describes the differences between the search progress data related to time, between the two conditions.

Table 7.36 compares the descriptives of the EvoBerry time progress data (EvPT1-3) and its Internet Explorer counterpart (IePT1-3). On average the EvoBerry condition shows higher times per question, an unsurprising result, given that EvT and IeT have already been identified as being significantly different in that respect. The range, minimum and maximum all vary

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvA1	2.85	3	3	0.489	2	1	3
IeA1	2.7	3	3	0.732	3	0	3
EvA2	2.7	3	3	0.732	3	0	3
IeA2	2.75	3	3	0.716	3	0	3
EvA3	0.55	1	1	0.510	1	0	1
IeA3	0.6	1	1	0.502	1	0	1

Table 7.37: **Data descriptives: EvPA - IePA.** This set of data describes the differences between the search progress data related to accuracy, between the two conditions.

Time	**Mean**	**Median**	**Mode**	**Standard deviation**	**Range**	**Min**	**Max**
EvSc1	4.777	4.5	6	2.734	9	1	10
IeSc1	5.473	4	2	4.005	14	1	15
EvSc2	2.722	2.5	5	1.637	5	0	5
IeSc2	3.105	3	1	2.024	6	1	7
EvSc3	2.111	2	1	1.323	5	0	5
IeSc3	3.368	2	3	4.336	19	0	19

Table 7.38: **Data descriptives: EvSc - IeSc.** This set of data describes the differences between the search progress data related to searches generated, between the two conditions.

greatly, and it should be noted that in both conditions, a common minimum and a common maximum is rarely shared (e.g. EvPT1 and IePT1 share similar ranges but, IePT1 has a lower minimum). Note that the mode is not available for several of the compared variables, since very few results appear more than once, often differing by one or two seconds. Two pieces of data are worth noting in the descriptives: (1) on average EvoBerry progress time is always greater than Internet Explorer by a factor of 2.9-3.0 minutes, a result that may prove to be significant, and (2) there is a steady decline in the means of both EvPT and IePT, with the time taken to answer each question dropping as time progresses. These will be discussed further in this section.

Table 7.37 compares the descriptives of the EvoBerry accuracy progress data (EvPA1-3) and its Internet Explorer counterpart (IePA1-3). Both conditions performed equally well (on average) for each question, with nearly exactly the same data averages, and levels of variance, which is unsurprising given the results of the EvA - IeA t-test (p-value = 0.880). The spread of points that can be scored per task (3 in the first two questions, and 1 in the last) is a very small range, and so deviations away from the mean are unlikely.

Table 7.38 compares the descriptives of the EvoBerry search progress data (EvPSc 1-3) and

its Internet Explorer counterpart (IePSc 1-3), showing that in terms of average searches performed, the EvoBerry condition seems to produce lower searches per question than the Internet Explorer condition. While the T-test (see Table 7.32) has shown the differences between EvSc and IeSc to not be significantly different, it is important to establish whether this holds true for the search progress data. Like the time progress data, there also seems to be a trend, of a decrease in the values over time. An interesting aspect of the data, is the difference in the variance between the two sets of data. In question 3, the range differs greatly (EvPSc3 = 5, IePSc3 = 19) which is strange given that both minimums are the same (zero), showing that at the later stages of the search, users in the Internet Explorer condition produce far greater amount and range of searches than their EvoBerry counterparts.

7.4.1 Analysis : search progress data

The descriptives identified two areas of interest in the search progress data, increases in test subjects efficiency over time, (in terms of task time and searches per task), and differences in test subjects performance when utilizing the different tools. In the first area of interest, the data is analyzed within individual conditions, and in the second area, the data is analyzed between the two conditions.

T-test analysis : within condition

A gradual decrease in both time taken to complete questions (EvPT, IePT) and number of searches per question (EvPSc, IePSc) throughout test subject's search sessions was noted. The differences between the second and third question results (in both time and searches) were expected, given that the second question required them to locate three pieces of information, while the third question required test subjects to locate only one piece of information. However both the first and second question required test subjects to locate three pieces of information, and a large difference in both time and searches was discovered between the two questions.

A t-test comparing the differences between the time taken to complete questions 1 and 2, and the searches taken to complete question 1 and 2, was performed and the differences between both sets of data proved to be significant (see Table 7.39). Combining this significant difference with the fact that the accuracy across both groups remains the same means that there was an increase in test subject's efficiency over time. Two theories were proposed to explain this increase in efficiency: that test subjects became more familiar with the tool's functions over time, and thus efficiency increased (i.e. a learning curve), or as information gathered increased, the need to search for new information decreased (because of recall), lowering the number of searches, and the time taken to complete questions. However, it was posited that if the effect was indeed a product of a learning curve, then it would not be present in the Internet Explorer

condition, given that test subjects are already intimately familiar with the tool.

T-test analysis : between conditions

The significance of the differences in the search progress data can be seen in Table 7.40. The t-test showed only one result was significantly different (EvPT3 - IePT3, p-value = 0.001). Given this evidence, the theory was posited that the significant difference in session times between the two conditions (EvT and IeT) is somehow related to the last stages of the search session (EvPT3 and IePT3). This discovery prompted an investigation into whether the time taken to complete the final question could be linked to an affecting variable, which in turn would help explain the time difference created between the two tools.

Pearson's correlation coefficient was applied to the data, in an attempt to establish a relationship between EvPT3 and the EvoBerry tool usage, as well as between IePT3 and Internet Explorer tool usage. A summary of the findings is presented in Table 7.41. A weak positive correlation was discovered between visual-history tool usage and the time taken to complete the third question in the EvoBerry condition. This correlation is supported by the previously posited theory that visual-history toolbar was affecting search progress, and implies that the visual-history toolbar played a part in increasing test subject's time in the later stages of the search scenario. The next section will move onto the experimental factors that may have affected the data.

7.5 Experimental factors

Certain aspects of the experimental design were not ideally implemented due to external constraints, and may have affected the experimental results. These aspects were sample size, exposure to the tool, and utilization of the tools.

Sample size

A large sample size is important for gathering accurate and significant results. As discussed previously, the lower the number of test subjects, the greater variance in the results and thus the lower the significance of the results received in the experiment. While it is possible to gather significant results with a test subject group of 20 subjects (as demonstrated by the experiment), the majority of differences in results were not significant. Given a larger test population, it is believed that a more significant set of results could be obtained.

The test subject population was set to twenty subjects for three reasons: (1) the experimental money allocated was only enough to pay 20 test subjects, (2) the time taken to conduct each experiment (minimum of 2 hours) constrained the amount of experiments that could be

EvoBerry data	**p-value**		*Internet Explorer data*	**p-value**
EvPT1 - EvPT2	0.002		IePT1 - IePT2	0.003
EvPSc1 - EvPSc2	0.005		IePSc1 - IePSc2	0.025

Table 7.39: **Search progress data : within conditions t-test** A t-test was performed to confirm the significance of the difference between the speed and number of searches taken to complete question 1 and question 2 (in both conditions). The t-test was a paired two-tail test, and all differences were seen to be highly significant.

compared variables	**p-value**
EvPT1 - IePT1	0.209
EvPT2 - IePT2	0.263
EvPT3 - IePT3	0.001
EvPA1 - IePA1	0.452
EvPA2 - IePA2	0.833
EvPA3 - IePA3	0.748
EvPSc1 - IePSc1	0.390
EvPSc2 - IePSc2	0.574
EvPSc3 - IePSc3	0.483

Table 7.40: **T-test results of the search progress data:** A two-tailed, paired t-test was performed on the comparison data to determine the significance of the differences in the results.

EvoBerry correlations	*r*		*Internet Explorer correlations*	*r*
EvPT3 - EvSb	0.143		IePT3 - IeFd	-0.306
EvPT3 - EvHs	0.502			
EvPT3 - EvBm	0.026			
EvPT3 - EvDb	-0.075			
EvPT3 - EvCo	0.086			
EvPT3 - EvFd	-0.333			

Table 7.41: **EvPT3 and IePT3 correlations.** This table shows the correlations of the time taken to complete the third question (in both conditions) correlated against the different tool usages. Note that in the Internet Explorer condition the only tool represented is the find tool, because no correlations could be established in either the bookmark or history tools given that their usages were 0 and 1 respectively.

conducted in the given time period, and (3) given the large experimental time, it was already difficult to find 20 willing participants (even with the monetary incentive). With a fixed amount of experimental money, the only way to pay for more test subjects would be to lower the monetary incentive per person, which is undesirable because it would drop the incentive to a rate that would be too low to attract test subjects. Lowering the experimental time is difficult when implementing an evolving search experiment, which requires a long and difficult search. Short of finding a method of performing an evolving search experiment in a lower time bracket, the only other option would be to change from a within-subjects experiment to a between-subjects experiment. This change would be undesirable because it would require twice as many test subjects to satisfy the same conditions, and would mean that any comparison of the results would be affected by individual differences between the test subjects.

In the area of psychology, experiments with test subject populations of fifty or more are required as a bare minimum to get significant results. However this is primarily due to the differences in structure and objectives between interface experiments and psychological experiments. For example, experiments testing a new interface, test subjects need to be given some training time in order to familiarize themselves with the interface, whereas psychological experiments which test human conditions often use innate tasks where no learning is required, thus removing the training time and lowering experimental time because they are using inborn skill sets. This in effect leads to a lower monetary pay per person, and allows the experiment designer more funds to secure more subjects. There are also cases where a psychological experiment will only utilize a handful of test subjects; in these cases significant results are obtained by repeating the experimental test several times. Depending on the time taken per task, this can sometimes be as large as 100 or 1000 times in the space of a few hours. However the current design for search result interface experiments means that this is not possible, evolving searches are long and often require information to be obtained *through-out the process* meaning that parts cannot be skipped. Thus repeating the experiment several times would not be ideal given that most scenarios have a duration of at least 30 minutes.

Tool exposure

Exposure to the tool is important because people are far more used to interacting with the Internet Explorer interface than the EvoBerry interface. A higher level of exposure to the EvoBerry tool, preferably over several sessions, is needed to obtain more comparable results. Two solutions to this problem, which were not applicable at the time of design were designing the tool as an add-on to an existing web browser and exposing test subjects to the tool over a longer period of time. The first solution was not applicable at the time, because the technology was not available to add the types of techniques in the design to a standard web browser such as

Internet Explorer. The second solution, exposing test subjects to the tool over longer periods of time, can be handled in two ways (both of which were not deemed suitable at the time). Firstly, the experiment could be carried out over the course of several days, so that the user's abilities with the tools could be monitored closely. This presented new problems, such as attracting test subjects who would be willing to sit through a 2 hour experiment over the course of several days. Similarly with restricted funding, it would not have been possible to pay users for the time they would spend over the several days, unless the number of test subjects used was lowered, which would have a knock-on effect lowering the significance of results and increasing the variance. The second method of exposing test subjects to the tool over a long period of time would be to install the tool locally on their systems, so that test subject's search behaviours can be monitored in day-to-day situations. However this too is fraught with problems. First, some systems are not compatible with all the tools used, for example the web browser renderer for EvoBerry is a Windows specific component, and would not work on either Mac OSX or any Linux system. Second, guaranteeing that test subjects use the tool for their daily searching would be difficult. The third problem is privacy; by recording test subject's search behaviour, every web page visited would be recorded, something that they may not wish.

Tool utilization

Tool utilization was low overall, with no tool in the EvoBerry interface being utilized by more than half the test subjects. However, analysis of tool usage shows that certain tools (such as the comparison tool and find tool) improve search efficiency, and thus, encouraging test subjects to utilize these tools may change the results considerably. However, getting the test subjects to utilize the various tools incorporated into the interface is difficult. Even if they enjoy using the tool it is often difficult to break out of old searching habits once they start concentrating on the task at hand. One way of ensuring that test subjects use the tools is to explicitly state that they should use specific tools in specific parts of the experiment, e.g. 'use the visual-history tool to recall page X'. The first problem is that it does not model a true evolving search, where users generally browse opportunistically when searching, and by specifying which actions to perform users natural search process is disrupted. The second problem is that certain tools have no directly comparable counterpart in Internet Explorer (such as the comparison tool), so the experimental questions would not be similar. Another method of ensuring that test subjects use a tool is by removing its equivalent versions, thus leaving test subjects no choice but to use the provided tool. For example removing the navigation buttons (backwards and forwards) in a web browser, and replacing it with the visual-history toolbar, would force test subjects to only use the history toolbar when navigating among web pages.

7.6 Overview

In this chapter the experimental results were presented and analyzed in detail. The results received differed from the initial hypotheses, but were nonetheless positive and interesting. Of the three main comparison variables, EvoBerry users were significantly slower than Internet Explorer users, but highly similar in accuracy, and with no significant differences in number of searches created. It was remarked that test subjects achieved the same level of accuracy with EvoBerry as with Internet Explorer, even though they were less familiar with utilizing the EvoBerry tool. The significant difference in time between the conditions was a concern, and identifying the factors which contributed to this difference, became the focus of the analysis.

In the comparison data, a weak correlation between the time taken to complete scenarios and the number of threads generated was identified. As thread generation increased so did time taken and thus it was theorized that whatever factor was increasing thread generation also increased the time taken by test subjects. It was posited that thread generation was linked to differences between the two interfaces, specifically the number of search results displayed. EvoBerry displayed a larger amount of results, and thus encouraged test subjects to spend more time exploring search results.

In the tool usage data, four tools were identified as having influenced search progress. The comparison tool was seen to have a positive influence on accuracy, and received a high amount of commendation (as detailed by the questionnaire data). The find tool was seen to have a positive influence on time taken to complete the scenario, but was under-utilized because it was not explicitly represented in the interface's display (and thus easily forgotten). It was posited that increased usage of both of these tools would improve search efficiency.

The session bar was seen to have a negative influence on accuracy, and was criticized for its small icon size and complexity. A relationship was established between the visual-history tool, and the number of web pages viewed, where increased usage of the tool would increase web pages viewed. It is difficult to determine whether the increased web page views was a positive effect (the tool encouraged more detailed browsing of web pages) or a negative effect (the tool hindered the search efforts forcing test subjects to open more web pages), although it should be noted that no negative comments were received regarding the tool.

In the search progress data, a significant difference was discovered (between the two conditions) in the time taken to complete the third question. It was posited that the difference in time between the two conditions may be related to whatever factors increased the time taken to complete the final question. A weak correlation was established between EvPT3 and visual-history usage, showing that visual-history tool may have played a part in increasing the test subjects time in the later stages of the search.

It is also possible that the experimental results were affected by external constraints, such

as the sample size, tool exposure and tool utilization. It was theorized that by increasing the number of experimental subjects a more significant set of results could be obtained. It was also believed that increased exposure to the interface, as well as encouraging test subjects to utilize the different tools of the EvoBerry interface, would improve performance, bringing the experimental results closer to the hypothesized results.

In summary, it was discovered that a combination of factors such as search behaviour, tool usage and experimental factors, all influenced the experimental results. The analysis also showed the effectiveness of the different tools in the EvoBerry interfaces, as well as identifying specific tools which were beneficial to test subjects, as well as tools which were detrimental to the search progress of test subjects.

Chapter 8

Conclusion

The focus of this research was to address the issues presented by the evolving search, and to design a search tool that would aid users in performing evolving searches. This final chapter will summarize the achievements in this work, discuss the challenges of the research, and look at the future direction of the research.

8.1 Achievements

In the process of investigating the visualization of evolving searches, significant contributions were made to research in this area. These contributions can be divided into three main categories: (1) theoretical work, investigating the background behind the evolving search model, (2) technical work, designing and implementing tools to aid users in evolving searches, and (3) experimental work, testing the efficiency of the tools developed.

Theoretical work

The evolving search model was originally designed around the interactions of users within libraries and with databases. Although, the model has been referenced with regards to web search behaviour (see, [43], [71]) neither investigation nor evidence had been put forward to support the existence of evolving searches in an online environment. Evidence was compiled supporting the existence of the evolving search as a web searching behaviour, based on observations of web search behaviour taken from various sources ([57], [59], [102]). Interface issues related to the evolving search (based on Bates [6] work) were noted and discussed. This was presented in chapter 1. A comprehensive investigation of the differences between the evolving search model and other theories and models related to web searching was presented. The investigation discussed the differences between information-retrieval and information-seeking search models, in

addition to presenting important information-seeking models in detail. It was determined that while some of the aspects of evolving search could be seen in these information-seeking models, other aspects remained unique to the evolving search model. This related work of search theories, was presented in chapter 2.

Technical work

Bates's [6] work identified several interface issues that needed to be addressed to enable users to perform more efficient evolving searches. These were combined with other interface issues, specific to information-seeking on the WWW, which were not present during the time of Bates's research. These issues formed the basis of the design specifications for the tools developed to aid users in their evolving searches. A comprehensive investigation of the different visualization technologies and tools that could aid users in performing evolving searches was presented. The current technologies used for web searching were also presented and discussed, as well as their limitations with regards to performing evolving searches. This related work of visualization techniques and technologies was presented in chapter 3. Tools that aided users in their evolving searches were designed, implemented and included in a set of published works. Three techniques/tools featured prominently: (1) the visual-bracketing technique (featured in [95]), which aided users in viewing large amounts of search results, (2) the search similarity tool (featured in both [110] and [111]), which aided users in seeking interesting and pertinent search results, and (3) the visual-history toolbar, which aided users in recalling specific web pages from their browser history. Each of these techniques was described and presented in chapter 4.

These works culminated in the design and implementation of the EvoBerry interface, an integrated search tool for aiding users in evolving searches. The EvoBerry interface (introduced in chapter 5) aided users in different aspects of their search. The visual-bracketing technique was applied to the display of the search results, and allowed users to move quickly and opportunistically over their data. Both the comparison tool and the similarity highlighting enabled users to find interesting and pertinent results by displaying the intersections of multiple searches.

Tools were provided for recalling search data. The session bar could be used to recall users web browsing sessions, and the visual-history toolbar allowed users to recall individual web pages. Both search result data and user inputted text could be stored in the drop-box, and search results of interest could be highlighted and returned to later. Chapter 5 presented an in-depth look at the different stages taken in the design of the tool, from the initial prototype, to the final version. The technical challenges behind designing such a tool were also discussed.

Experimental work

In order to evaluate the effectiveness of the EvoBerry interface and its various tools, an information-seeking experiment was designed and implemented using 20 test subjects. A within-subjects experimental design was used, and each of the subjects was tasked with completing two search scenarios, one using the EvoBerry tool, the other using Internet Explorer. An automatic logging program recorded all of the users interactions with each of the tools and buttons in the EvoBerry tool. In addition to performance data (time, accuracy and number of searches generated per user), various other variables related to the performance of users (such as number of search results opened, and web pages viewed) were also recorded. Screen capture videos were made of each user's search sessions, and over 20 gigabytes of screen-capture videos were gathered. These videos enabled us to observe users search patterns and behaviours in detail. User's opinions on the EvoBerry interface and its individual tools were gathered through the use of a post-experiment questionnaire. Both the design of the experiment and the results obtained were presented in chapter 6.

An analysis of the results was presented in chapter 7. When comparing the performance of users in the two conditions (EvoBerry and Internet Explorer) it was discovered that the accuracy of users was very similar between the two conditions. Although different to the expected hypothesis, this is seen as a positive outcome, showing that users were able to perform as efficiently with the EvoBerry tool (a tool which the test subjects have had little training or exposure to) as with the Internet Explorer tool (a tool that all test subjects have used extensively). A correlation was also made between users accuracy and usage of the comparison tool. Of all the tools that formed the EvoBerry interface, the comparison tool was the most highly praised (as evident in the questionnaire data) and the most often used of the tools (as evident in the tool-usage data). Given this evidence, it was determined that the similarity tool was a valuable asset in aiding users in performing evolving searches. The analysis also helped identify parts of the EvoBerry tool which did not contribute positively to the search progress of users (such as the visual-history toolbar and session bar) providing information which benefit future design's of evolving search tools.

8.2 Challenges of the research

Much was learned through the investigation, development and experimentation of evolving searches, and the resulting work has added to the current research in the area. However, various challenges and issues arose over the course of the work, many of which were discussed earlier in this thesis. This section will review the main challenges, as well as the rationale behind the solutions chosen.

Challenge I : issues addressed

The first challenge arose early in the research, and involved choosing which interface issues were to be addressed in the work. Several interface issues were presented in Bates's work, but not all of these issues were applicable to the research. This was due to the fact that Bates's theories were modeled around the interactions of users with library services and databases, thus were not all applicable to searching on the WWW. Some interface issues could not be accurately mapped to web searching. For example, one of Bates's interface issues provides the ability to switch between multiple search strategies, however some of the strategies that she lists, such as area scanning, cannot be replicated in a web search interface. In the case of area scanning, information on the WWW is not neatly ordered and classified like books on a library shelf, thus related web resources are rarely located physically close together, and cannot be browsed as easily as scanning a shelf of books.

Only three of Bates's six interface issues were addressed in this work, and were supplemented with two other interface issues, 'information-seeking' and 'information management'. These interface issues were not mentioned by Bates because they were problems that did not exist in the searching of library or database services. The rationale behind both information-seeking and information management was that users should be provided with the means to respectively, seek and manage data efficiently, due to the large amounts of information generated while performing evolving searches on the WWW.

Challenge II : tool design

The second challenge was related to the design of the tool. The Java programming language was used to implement the tool, and provided a very flexible set of tools for designing the interface. However, certain aspects of Java, crucial to the design of the EvoBerry interface, have not yet advanced to the point where the interface's functions can be fully realized. The aspect of concern is the Web Browser object utilized in the EvoBerry tool, developed by the JDIC project [23]. The JDIC web Browser was utilized because (at the time), it was the only Java package that could render a web page exactly as displayed in a web browser, within a Java program. However, the JDIC web browser was not ideal because it was designed as a Java heavy-weight program, and thus did not interact well with the JDesktopPane interface utilized, which was a Java light-weight component. This resulted in the use of a JSplitPane to contain the Web Browser object.

During the experiment, users complained that the JSplitPane was inconveniant to interact with, because it was not contained within a JInternalFrame, and could not be moved around freely on the desktop. This resulted in users having to constantly resize the split-pane when switching between viewing search results and browsing web pages. In addition to this, sometimes

new searches were generated 'beneath' the split-pane, and were lost to users. The only alternative to utilizing a JSplitPane was to open web pages in an external web browser, which was not considered a viable option, given that users would then be forced to constantly switch between the EvoBerry window, and the web browser windows. The split-pane was not ideal for the EvoBerry tool, and will be replaced once Java technology advances to the point where web browser objects can exist within JInternalFrames.

Challenge III : experimental design

The third challenge was related to the design of the EvoBerry experiment, and was linked to the fact that evolving searches are both long and difficult to complete. An evolving search takes a long time to complete because users often begin with an incomplete information need. Either they have not identified a particular piece of information to pursue (e.g. are browsing opportunistically) or do not have enough personal knowledge to form an effective search engine query. Attempting to simulate an EvoBerry search accurately within an experiment can cause problems, because of the large amount of time required to complete the search. Experimental subjects are paid a monetary reward based on the amount of time they spend performing the experiment, but in experiments with long durations, such as evolving search experiments, higher amounts of money must be paid out per subject, thus lowering the amount of test subjects that can be gathered. Less test subjects means a lower chance of obtaining significant results.

Even without the issue of monetary reward, the time taken to complete the experiment may act as a deterrent to potential test subjects, who are too busy to spend such a large amount of time in an experiment. The large experimental time also affects the experimental results, since the motivation of users deteriorates as time progresses. In order to more accurately compare performance differences between two tools, the same users should perform similar tasks using both tools, i.e. a within-subjects experimental design should be used. However, in the case of an evolving search experiment this would double the already long experimental time, since users have to perform two sets of tasks instead of one. While this leads to more accurate results, it also makes it more difficult to get a large number of test subjects.

The three challenges presented in this section had different effects upon the research and the experiments conducted. Focusing on only a select set of interface issues concentrated the research in a particular direction, leaving other areas of the evolving search neglected, but open to future research. Placing the web browser object within a JSplitPane was not an ideal solution and inconvenienced users, but was considered a better alternative to using external web browser windows. There are plans to change this design in future implementations. The long experimental times associated with evolving search experiments will continue to be troublesome

in future experiments, and perhaps encouraging users to utilize the EvoBerry interface for their web searches would be an easier method of observing user performance.

8.3 Future work

The research presented, as well as the experiment conducted in this work are merely the tip of the iceberg, and there is still much work to be done in the area of evolving searches. The future of this work can take many directions: (1) extending and improving the current designs and experiments, (2) investigating other evolving search issues, not covered in this work, and (3) following new lines of research inspired by the results received from the experiment.

8.3.1 Extending the current design

The current design can be improved upon in two ways: (1) by fixing flaws in the original design, and (2) using new technology to overcome constraints in the original design. The results analysis showed that some tools, such as the session bar, created a negative effect upon users performance, necessitating the need to remove or redesign such tools for future versions. It was also discovered that some tools enhanced user's search performance, such as the find tool, and as a result should be made more prominent in future versions of the tool.

The design of the EvoBerry tool suffered from a few technical constraints (such as having a fixed web browser) which, through improvements in technology, may no longer be an issue. The number of different Web APIs has increased greatly over the last 3 years, and the services have also improved and diversified. New services are now available, and searches of images, audio and video media can now be seen as part of the standard Web API services. The current EvoBerry design is suitable for research purposes, however deeper integration of the EvoBerry tools within current web browsers is needed. This can be done in one of two ways, either by embedding the tools directly into web browsers (as either plug-ins or toolbars), or generating the tool as an 'in-page' interface (such as the current web search engines) with the tools embedded into a web page using Java Servlets or Macromedia Flash technology.

By embedding the EvoBerry tools into the web browser, the portability and usability of the tools will be increased, as well as the acceptability of such tools by users. Inclusion of the EvoBerry tools as a plug-in would also create a more comparable interface for future experiments, and would potentially allow longer exposure of the tool to users. Embedding the tools into the web page itself removes the limitations of browser-dependent plug-ins, and makes the tools instantly accessible to any person in the world. However, unlike browser plug-ins, the performance of the tools would also be dependent on bandwidth, and as a result (given the complexity of some of the EvoBerry tools) the performance of the tools would suffer greatly

over connections with a low bandwidth. Storing and accessing user's personal information (such as past searches, and web pages viewed) would also become more troublesome, since the tool (and its data) are not stored locally.

These changes in technology also affect the way that experiments will be conducted. Moving to an embedded design allows easier access, and in turn more wide-spread availability of the tool. This is beneficial to experimental design in several ways: (1) increasing the availability of the tool increases users exposure to the tool, and in turn gives them more experience in using the tool making for a more accurate experiment, (2) increasing the availability of the tool increases the potential test subject population, which in turn will increase the significance of any results obtained, and (3) using a plug-in would allow a more accurate comparison of the tool with a standard web browser (sans the EvoBerry plug-in).

8.3.2 Investigating new issues

This research has only looked at 3 of the 6 interface issues reported by Bates [6] in her work. The research could be expanded to encompass the remaining three issues detailed by Bates, although different designs and techniques would need to be utilized, perhaps requiring the development of a separate tool. These interface issues are presented.

Many of the different *strategies* described by Bates were based on searching in a library, and thus not applicable to web searching. However, research and our own observations have shown that web searching (as well as the evolving search) does not consist of one search strategy, but instead a variety of different strategies applied at different parts of the search, and in different situations. Investigating the different search strategies, as well as the design of tools to support them may prove to be a fruitful area of research.

Bates theorized that designing an new interface around an analogy would aid users in understanding its functions. The analogy posited by Bates was that of a library, where users could 'browse' the books on a shelf, and 'flip' through the pages of the books. Perhaps, other analogies could be used in the design of tools, for example, a 'berry-picking' interface, where users moved between various 'bushes' (searches), inspecting and selecting 'berries' (search results). As technologies change and improve, the amount and sources of information available increase, and as a result so does the range of *information-switching* that can take place. A larger number of Web APIs are now available, with even greater numbers of new services, allowing tool designers to draw on information from several different search engines, as well in several different forms of data. Indeed, future technologies may enable access to specialized searches of specific websites (e.g. Amazon.com, Ebay.com) allowing a greater degree of information-switching.

8.3.3 Following new research

The research has identified *search result comparison* as a useful technique when performing web searches, and has prompted further investigation into its application within the online searching environment, as well as other areas of searching. Given the results received from the experiment, a more detailed investigation of the impact of search result comparison is needed. The creation of a web browser plug-in, based on the design of the comparison tool, is being considered. This would allow a more focused and comparable experiment, which would more clearly show the performance difference created by the tool. The research performed was limited to the comparison of searches by a single user in a current search session. It would be interesting to examine the effects of extending this time-frame, observing the effects of allowing users to compare searches separated by days, months or even years. Examining the effects of allowing users to compare their searches with that of different users may show interesting results.

The research also restricted users in the type of information being compared as well, as well as the programs interpretation of 'similarity' between different pieces of information. The comparison principle can be extended to other pieces of metadata (images, keywords), as well as other rules of similarity (e.g. web pages containing the following keywords...). The comparison of multiple searches is not a technique that is solely restricted to searching in the online environment. Search result comparison (and indeed the principles of the evolving search) can be extended to other domains, where users must search through a large amount of data with only a vague information need, e.g. searching through old emails or files on your hard drive.

8.4 Overview

This thesis has investigated various aspects of the evolving search, and provided a solution to enable users to perform and manage their own evolving searches. A comprehensive study was performed, and demonstrated that the techniques and ideas utilized were useful. While beneficial components of the system were indeed identified, further experimentation needs to be carried out in order to determine the full effectiveness of the tool; in particular increasing users exposure to the tool over a longer period of time is necessary.

This thesis has generated new ideas which can be further developed and expanded upon. There is more research to be done in the area of *search result comparison*, and the comparison of multiple searches. Extending the principles of search result comparison to searches separated by different times (days, weeks, months), users, or programs may be beneficial.

Furthermore, the principles of the evolving search are not confined to the WWW alone. The issues of (1) starting a search with a vague information need, and (2) having to search through large amounts of data, are not unique to searching on the WWW, and can be seen in

searches of emails, documents, spreadsheets etc. Hence the technologies, techniques and ideas that have been presented in this thesis, especially those to manage users in their exploration and comparison of results between searches, can easily be adapted, extended and usefully applied to a great number of other domains.

Appendix A

Experimental hand-outs

This part of the appendices contains copies of the three hand-outs given to users during the experiment. The pre-experiment questionnaire was used to screen users and determine their web searching history, as well as their experience with Evolving searches. The EvoBerry guide was given before the experiment, and served as a reference guide to the user, when learning how to use the tool, as well as during the experiment. The post-experiment questionnaire, was given to the user after the experiment in order to gauge the users opinions on the EvoBerry interface and its tools.

Evolving Search Experiment
Preliminary Questionnaire

Do you use search engines to find information on the Internet?

Yes ☐ **No** ☐

Which search engines have you used?

Alta Vista	☐	**Dog Pile**	☐
Google	☐	**MSN search**	☐
Yahoo	☐	**Ask Jeeves**	☐
Vivisimo	☐	**Grokker**	☐
Other	☐		

Which search engine(s) do you use most regularly?
(Ranked by usage, 1 being highest)

1.)	
2.)	
3.)	

Please turn over.

Figure A.82: **Pre-experiment questionnaire page 1.**

How often do you use a search engine?

Once a month	
Once a Week	
2-3 Times a week	
At least Once a day	
More than once a day	

Have you ever used a search engine to:

To do research for your Work/Job	
To hunt down research papers	
Look for presents and gifts	
Plan a trip or a holiday abroad	

Figure A.83: **Pre-experiment questionnaire page 2.**

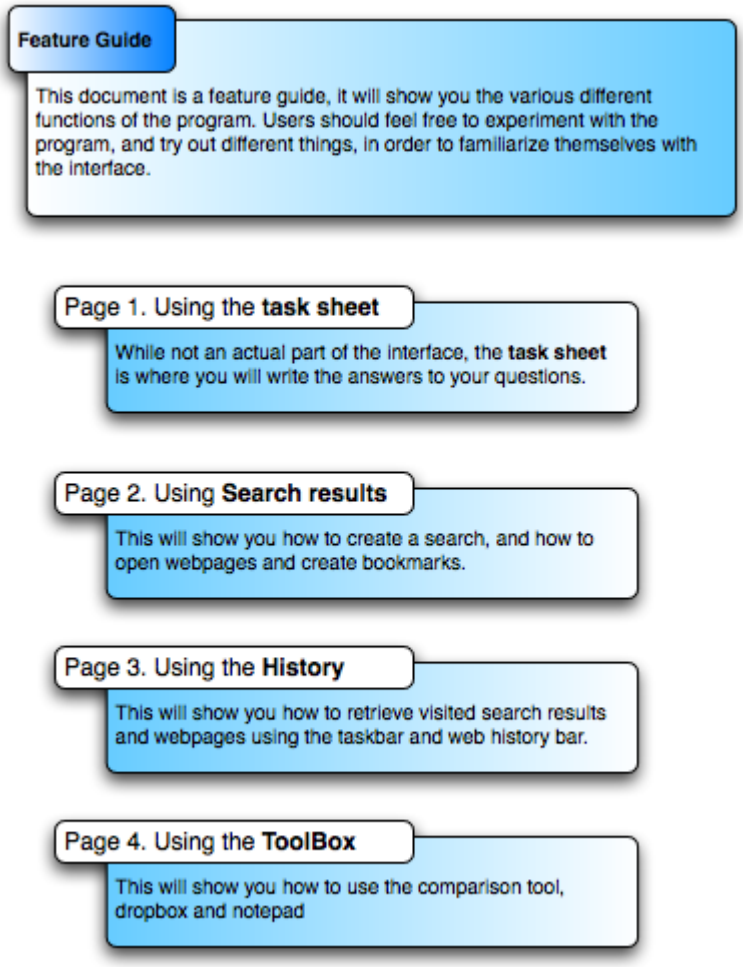

Figure A.84: **EvoBerry guide page 1.**

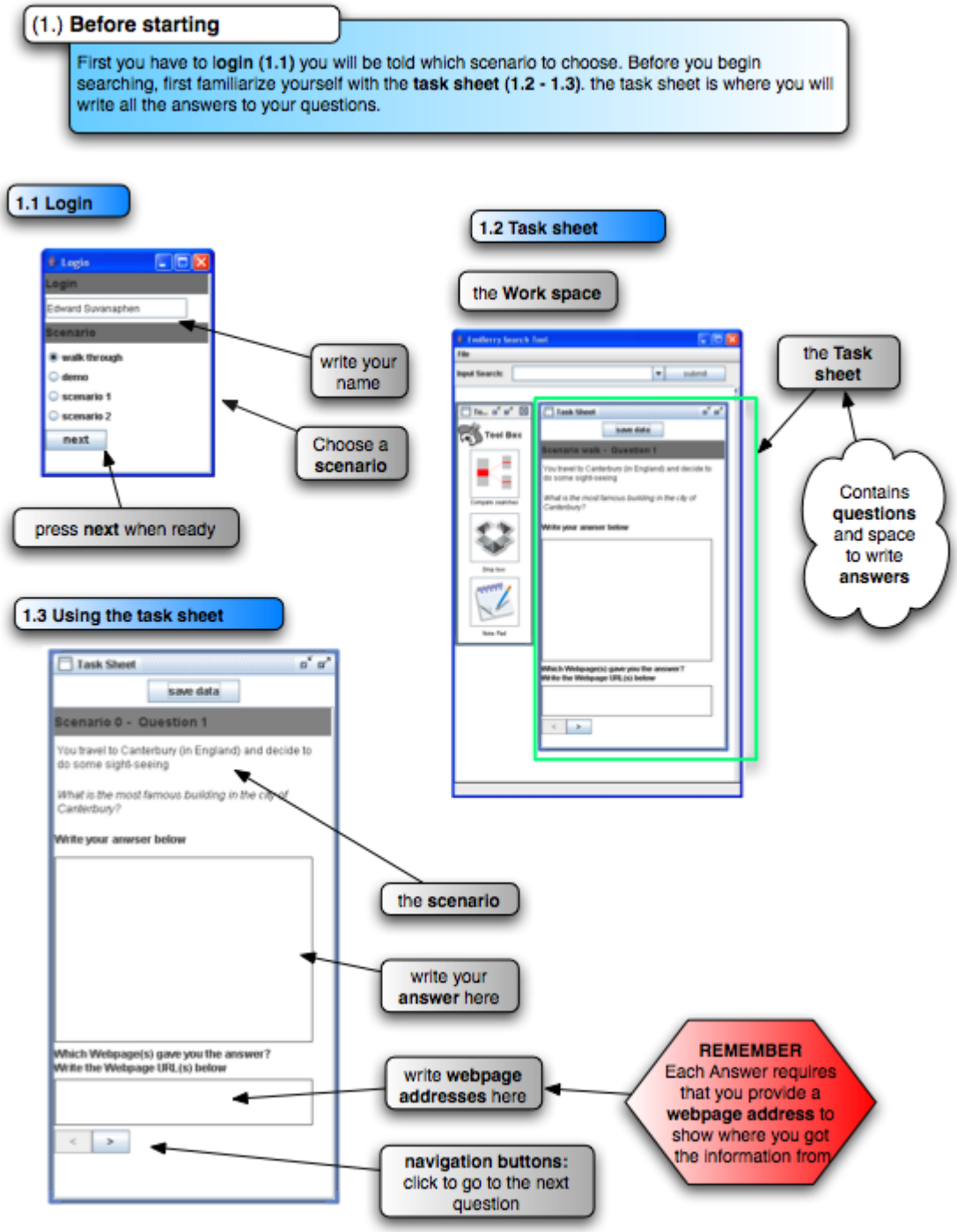

Figure A.85: **EvoBerry guide page 2.**

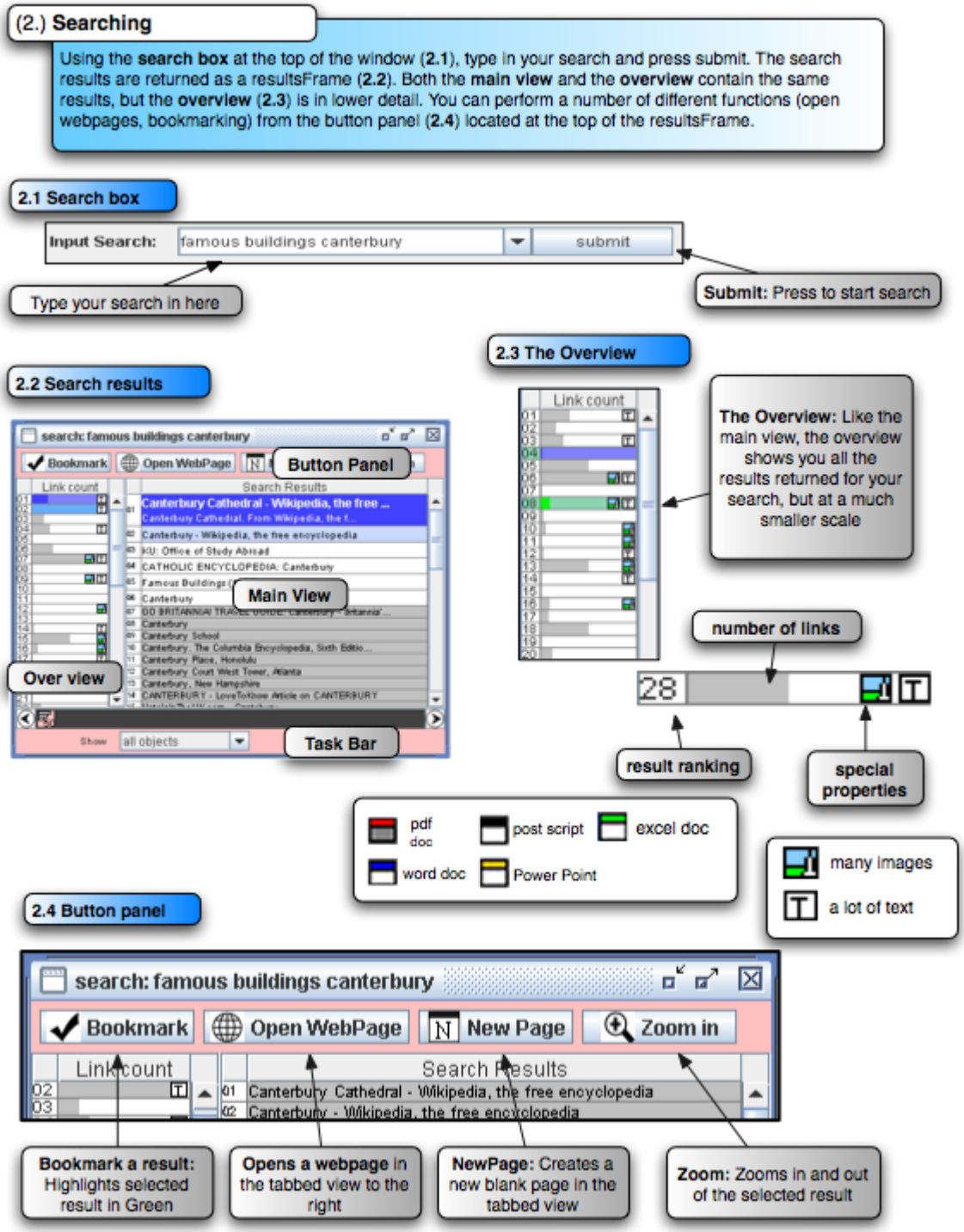

Figure A.86: **EvoBerry guide page 3.**

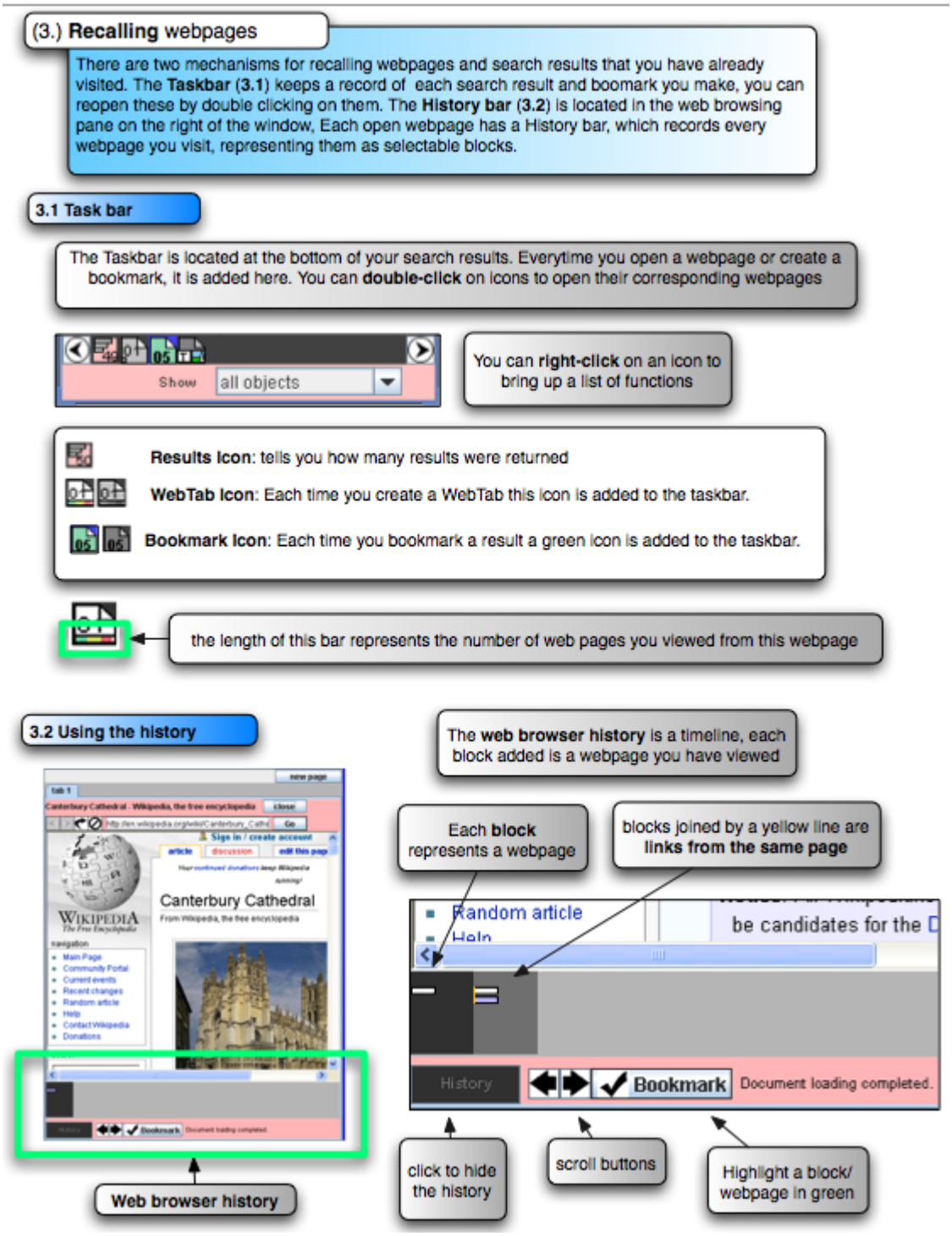

Figure A.87: **EvoBerry guide page 4.**

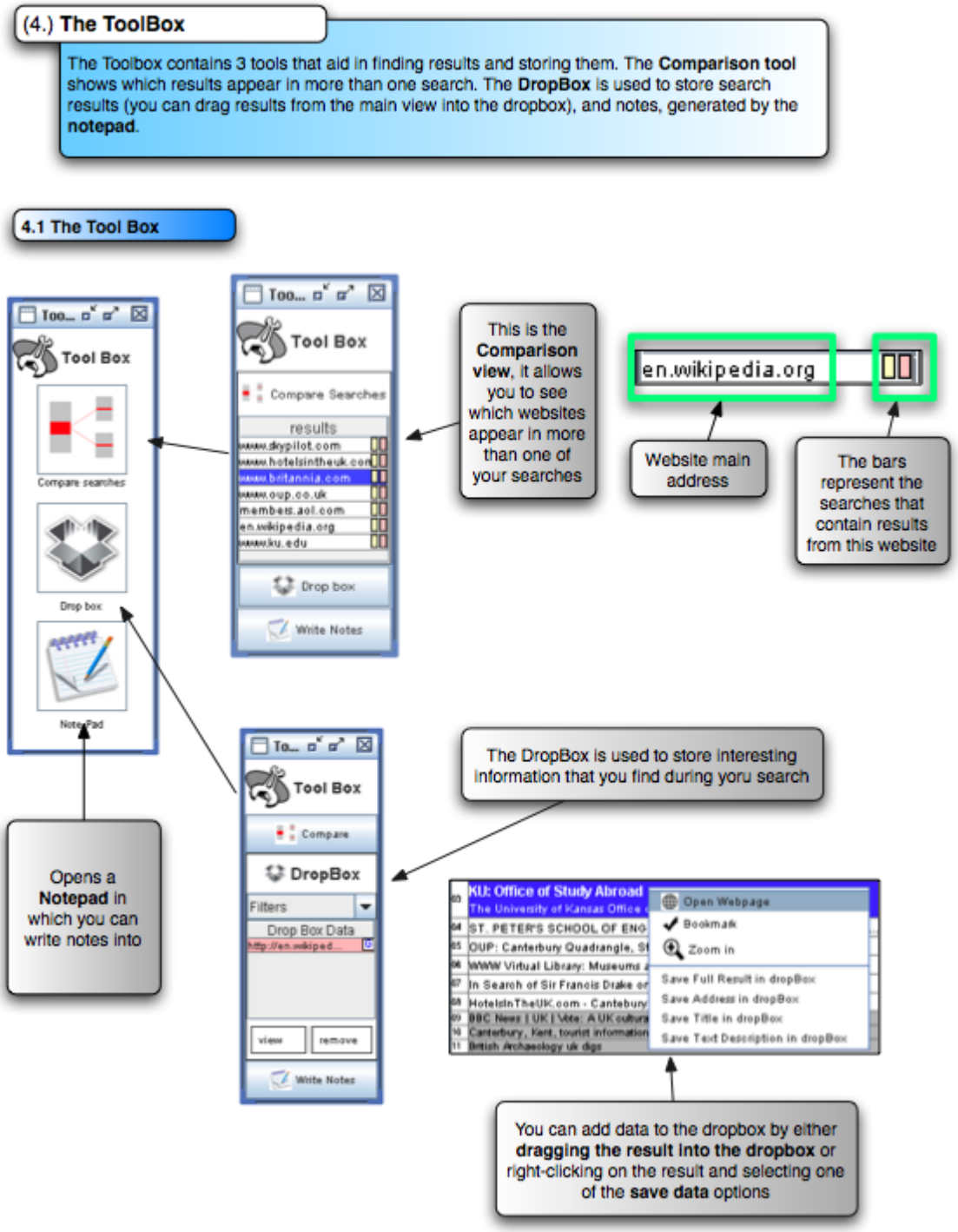

Figure A.88: **EvoBerry guide page 5.**

Evolving Search Experiment
Post-Experiment Questionnaire

Place a cross on the line to indicate your answer.

How often did you use the Book marking function?

Very Often	_____	Never

How often did you use the Comparison tool?

Very Often	_____	Never

How often did you use Visual history function?

Very Often	_____	Never

Were the instructions in the Walkthrough easy or hard to understand?

Very Easy	_____	Very Hard

Comments:

Was the Comparison tool easy to use?

Very Easy	_____	Very Hard

Comments:

Figure A.89: **Post-experiment questionnaire page 1.**

Was the Visual history function easy to use?

Very Easy		Very Hard

Comments:

As a whole, was the Tool easy or hard to use?

Very Easy		Very Hard

Comments:

Was there anything that you didn't like or found confusing about the visualization?

Any other comments (e.g. Any features you particularly liked?)

Figure A.90: **Post-experiment questionnaire page 2.**

Appendix B

Logged and observed data

This part of the appendices contains all the data compiled from the automatic logs and observations of the test subject's screen capture videos. Note that different systems were used to capture different pieces of data, the main time and and accuracy data was always recorded from the answer sheet, while the number of searches, threads, and web pages were captured through observations of the screen captures. All of the tool-usage data and some of the search progress data was also recorded through observations of the screen capture videos. A legend is provided with a recap of the names of the different variables of the data recorded in Table B.42.

Corrupt data

Three out of the Forty videos were found to be corrupted, and thus some of the data is missing. In cases where data is a compared, then data which cannot be paired is ignored. For example, When performing a paired t-test on EvSc and IeSc, the data for users A01, A07 and A11 were ignored since they cannot be paired. If a piece of data in a table is corrupted, then an X is placed within the box. Note that that none of the accuracy or time data was corrupted, since this was recorded by the answer sheet programs automatic logging program. The three corrupt videos were:

- **A01-EvoBerry session video:** In the comparison data this affected the variables, EvSc, EvTh, EvPg. In this search progress data, EvPT1-3 and EvSc1-3 were affected. No tool usage data for EvoBerry was recorded.
- **A07-EvoBerry session video:** In the comparison data this affected the variables, EvSc, EvTh, EvPg. In this search progress data, EvPT1-3 and EvSc1-3 were affected. No tool usage data for EvoBerry was recorded.

- **A11-Internet Explorer session video:** In the comparison data this affected the variables, IeSc, IeTh and IePg. In the search progress data, IePT1-3 and IeSc1-3 were affected. No tool usage data for Internet Explorer was recorded.

	Legend	
Comparison data	**EvoBerry**	**Internet Explorer**
Accuracy	EvA	IeA
Time	EvT	IeT
Searches	EvSc	IeSc
Threads	EvTh	IeTh
Web pages	EvPg	IePg
Search progress data	**EvoBerry**	**Internet Explorer**
Accuracy	EvPA (1-3)	IePA (1-3)
Time	EvPT (1-3)	IePT (1-3)
Searches	EvPSc (1-3)	IePSc (1-3)
Tool usage data 1 : Both EV & IE	**EvoBerry**	**Internet Explorer**
History	EvHs	IeHs
Find	EvFd	IeFd
Bookmark	EvBm	IeBm
Tool usage data 2 : Only EV	**EvoBerry**	
Session bar	EvSb	
Drop-box	EvDb	
Comparison tool	EvCo	

Table B.42: **Legend of variables in the experimental data.**

	time		acc		sch		thr		pg	
P	*EvT*	*IeT*	*EvA*	*IeA*	*EvSc*	*IeSc*	*EvTh*	*IeTh*	*EvPg*	*IePg*
A01	45	22	4	7	X	13	X	12	X	52
A02	48	19	7	7	13	5	25	9	57	41
A03	43	19	7	6	14	6	30	13	43	40
A04	33	24	6	7	7	5	31	19	76	65
A05	42	40	6	7	5	8	17	22	23	39
A06	22	32	7	7	7	27	12	21	61	67
A07	47	37	6	7	X	25	X	30	X	97
A08	34	13	6	6	7	4	12	14	55	19
A09	52	60	7	4	10	18	25	45	113	179
A10	36	21	6	6	11	7	37	12	65	33
A11	33	29	6	5	16	X	16	X	32	X
A12	32	19	5	3	10	14	22	18	34	20
A13	33	44	7	7	13	20	14	26	33	106
A14	34	15	5	7	15	9	16	11	27	21
A15	41	18	6	6	14	13	56	10	85	29
A16	30	15	6	7	8	11	24	11	26	32
A17	32	31	6	6	12	8	9	15	32	26
A18	42	40	5	6	5	9	18	11	65	62
A19	31	40	7	5	4	9	14	17	49	71
A20	17	19	7	5	5	11	10	12	33	42

Figure B.91: **Combined comparison data.**

	EvoBerry								
	ACC			TIME			SCH		
	EvPA1	*EvPA2*	*EvPA3*	*EvPT1*	*EvPT2*	*EvPT3*	*EvSc1*	*EvSc2*	*EvSc3*
A01	3	0	1	X	X	X	X	X	X
A02	3	3	1	24.4	15.1	8.52	6	5	2
A03	3	3	1	22	15.45	5.09	10	3	1
A04	3	3	0	16.46	7.73	8.21	4	1	2
A05	3	3	0	18.44	10.8	12.93	1	1	3
A06	3	3	1	14.55	4.79	2.92	6	1	0
A07	3	3	0	X	X	X	X	X	X
A08	3	3	0	13.57	8.45	11.22	3	3	1
A09	3	3	1	36.7	0.1	15.78	6	0	4
A10	3	3	0	14.37	10.66	10.97	4	5	2
A11	3	2	1	13.57	14.83	4.97	9	4	3
A12	3	2	0	11.27	11.03	9.7	4	2	4
A13	3	3	1	22.12	6.13	12.88	5	3	5
A14	1	3	1	15.01	9.02	9.27	6	5	2
A15	3	3	0	25.15	8.89	6.26	9	4	1
A16	3	3	0	14.17	7.12	7.92	6	2	3
A17	3	2	1	12.09	12.96	4.18	2	5	1
A18	2	3	0	16.35	15.9	10.34	2	2	1
A19	3	3	1	19.37	6.14	5.56	2	1	1
A20	3	3	1	5.36	8.86	4.01	1	2	2

Figure B.92: **EvoBerry search progress data.**

Internet Explorer

ACC			TIME			SCH		
IePA1	*IePA2*	*IePA3*	*IePT1*	*IePT2*	*IePT3*	*IePSc1*	*IePSc2*	*IePsc3*
3	3	1	10.3	5.2	3.00	4	6	4
3	3	1	15	3.12	1.35	4	1	0
3	2	1	12.1	3.14	4.21	2	1	3
3	3	1	12.56	8.45	3.05	2	3	1
3	3	1	31.2	2.8	5.40	6	1	1
3	3	1	18.08	10.39	3.63	15	7	5
3	3	1	13.04	2.25	22.08	5	1	19
3	3	0	4.43	3.8	4.11	2	1	1
3	0	1	33.52	10.98	16.68	12	4	2
3	3	0	8.3	8.7	4.29	3	2	2
3	2	0	X	X	X	X	X	X
0	3	0	10.45	6.75	2.26	6	5	3
3	3	1	14.21	12.27	15.65	10	2	8
3	3	1	2.46	7.67	5.27	1	6	2
3	3	0	11.12	2.46	4.92	7	3	3
3	3	1	2.3	4.85	1.15	1	3	6
3	3	0	26.05	2.25	2.75	11	1	0
2	3	1	28.15	9.25	2.77	7	2	0
2	3	0	13.15	21.15	4.84	3	5	1
2	3	0	9.45	3.99	5.83	3	5	3

Figure B.93: **Internet Explorer search progress data.**

	EvoBerry						IE		
P	*EvSb*	*EvHs*	*EvBm*	*EvDb*	*EvCo*	*EvFd*	*IeFd*	*IeHs*	*IeBn*
A01	X	X	X	X	X	X	0	0	
A02	5	2	0	7	1	0	11	0	
A03	0	0	8	7	0	0	4	0	
A04	3	0	0	23	3	3	0	0	
A05	0	0	3	2	0	0	1	0	
A06	0	0	0	0	10	0	1	0	
A07	X	X	X	X	X	X	0	0	
A08	0	3	3	0	0	0	0	0	
A09	0	29	0	3	3	1	0	0	
A10	0	0	0	0	0	0	4	0	
A11	1	0	0	0	2	2	X	X	X
A12	1	0	3	7	0	0	3	0	
A13	0	0	0	0	26	0	0	0	
A14	6	0	2	0	1	3	2	0	
A15	0	0	0	5	1	11	1	0	
A16	2	2	0	0	1	1	0	0	
A17	0	0	1	0	0	0	1	0	
A18	25	0	3	0	5	4	0	0	
A19	0	5	0	0	16	1	7	1	
A20	0	0	0	7	1	30	0	0	

Figure B.94: **Combined tool usage data.**

Appendix C

Miscellanous

Some data, when in a printed format is far too vast in quantity to fit in the thesis. Below are two of the sets of data could not be fit into the thesis, but are available upon request.

C.1 Questionnaire data

Because of the extra eighty pages that would be needed to accommodate the post-questionnaire data, this data was not included within the thesis. However, photocopies of the user's post-experiment questionnaires are available upon request. The results of the user's post-experiment questionnaire can be seen in chapter 6.

C.2 EvoBerry tool java code

The program spans over 100 classes, some of which are significantly large. If you wish to receive a copy of the code, and instructions on how to setup the program on your computer, email me at **es45@kent.ac.uk**, and I will make the code available for download.

Bibliography

[1] Anand Agarawala and Ravin Balakrishnan. Keepin it real: Pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of CHI 2006*, pages 1283–1292. ACM Press, 2006. http://citeseer.ist.psu.edu/761044.html.

[2] Christopher Ahlberg and Ben Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *CHI '94: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 313–317, New York, USA, 1994. ACM Press.

[3] Google Web API. Soap search api. (website), August 2006. http://www.google.com/apis/.

[4] Eric Z. Ayers and John T. Stasko. Using Graphic History in Browsing the World Wide Web. In *Proceedings of 4th International World Wide Web Conference (WWW4)*, pages 259–270. O'Reilly publishing, December 1995. http://www.citeseer.ist.psu.edu/195150.html.

[5] David J. Barnes, Mark T. Russell, and Mark C. Wheadon. Developing and adapting UNIX tools for workstations. In *Proceedings of the European UNIX Users Group (EUUG)*, pages 321–333. EUUG, Buntingford, UK, 1988.

[6] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online review*, 13(5):407–424, 1989.

[7] Patrick Baudisch, Bongshin Lee, and Libby Hanna. Fishnet, a fisheye web browser with search term popouts: a comparative evaluation with overview and linear view. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 133–140, New York, USA, 2004. ACM Press.

[8] Nicholas J. Belkin. Interaction with texts: Information retrieval as information-seeking behavior. In *Information Retrieval*, pages 55–66. Universitaetsverlag Konstanz, Von der Modellierung zur Anwendung, 1993. http://www.citeseer.ist.psu.edu/belkin93interaction.html.

[9] Nicholas J. Belkin, Colleen Cool, Adelheit Stein, and Ulrich Thiel. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. In *Expert Systems with Applications*, volume 9(3), pages 379–395. Elsevier Science Publishers Ltd., 1994. http://www.citeseer.ist.psu.edu/belkin94case.html.

[10] Nicholas J. Belkin, Robert N. Oddy, and Helen M. Brooks. Ask for information retrieval: Part 1. background and theory. In *Journal of Documentation*, volume 38(2), pages 61–71. Emerald Group Publishing Limited, 1982. http://www.scils.rutgers.edu/ belkin/articles/.

[11] Scott Berkun. How to build a better web browser. (online article), 2004. http://www.scottberkun.com/essays/essay37.htm.

[12] Alan F. Blackwell, Kirsten N. Whitley, Judith Good, and Marian Petre. Cognitive factors in programming with diagrams. In *Artificial Intelligence Review*, volume 15(1/2), pages 95–114. Springer Netherlands, 2001. http://www.citeseer.ist.psu.edu/375325.html.

[13] Christine L. Borgman. Why are online catalogs still hard to use? In *Journal of the American Society for Information Science*, volume 47(7), pages 493–503. John Wiley and Sons, New York, 1996.

[14] Kevin W. Boyack, Brian N. Wylie, and George S. Davidson. Domain visualization using vxinsight for science and technology management. In *Journal of the American Society for Information Science*, volume 53(9), pages 764–774, New York, USA, 2002. John Wiley & Sons, Inc.

[15] Auto Bracketing. (website), august 2006. http://www.dpreview.com/.

[16] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. In *Computer Networks and ISDN Systems*, volume 30(1), pages 107–117. Elsevier Science Publishers Ltd., 1998. http://www.citeseer.ist.psu.edu/brin98anatomy.html.

[17] Fidel Cacheda and Ángel Viña. Experiencies retrieving information in the world wide web. In *ISCC '01: Proceedings of the Sixth IEEE Symposium on Computers and Communications*, page 72, Washington, DC, USA, 2001. IEEE Computer Society. http://www.citeseer.ist.psu.edu/cacheda01experiencies.html.

[18] Stuart K. Card, Lichan Hong, Jock D. Mackinlay, and Ed H. Chi. 3book: A 3d electronic smart book. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 303–307, New York, USA, May 2004. ACM Press.

[19] Stuart K. Card, George G. Robertson, and William York. The webbook and the web forager: An information workspace for the world-wide web. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'96*, pages 416–417. ACM press, 1996. http://www.citeseer.ist.psu.edu/card96webbook.html.

[20] Chun Wei Choo, Brian Detlor, and Don Turnbull. Information seeking on the web: An integrated model of browsing and searching. *First Monday (online article)*, 5(2), 2000. http://www.firstmonday.org/issues/issue5_2/choo/index.html.

[21] Clusty. The clustering search engine. (website), August 2006. www.clusty.com.

[22] Andrew Cockburn and Steve Jones. Which way now? analysing and easing inadequacies in www navigation. In *International Journal of Human Computer Studies*, volume 45, pages 105–129. Academic Press Inc, 1996. http://www.citeseer.ist.psu.edu/cockburn00which.html.

[23] JDIC (JDesktop Integrated Components). Web browser project, August 2006. https://jdic.dev.java.net/.

[24] John Cugini, Sharon Laskowski, and Marc Sebrechts. Design of 3-d visualization of search results: Evolution and evaluation. In Robert F. Erbacher; Philip C. Chen; Jonathan C. Roberts; Craig M. Wittenbrink, editor, *SPIE*, volume 3960, pages 198–210. SPIE–The International Society for Optical Engineering, February 2000. http://www.citeseer.ist.psu.edu/323698.html.

[25] Richard L. Daft and Karl E. Weick. Toward a model of organizations as interpretation systems. In *Academy of Management Review*, volume 9, pages 284–295. Academy of Management, 1984.

[26] Fergus Daly, David J. Hand, Chris Jones, Daniel Lunn, and Kenvin McConway. *Elements of statistics*. Addison-Wesley, 1997.

[27] Bert J. Dempsey, Robert C. Vreeland, Robert G. Sumner Jr., and Kiduk Yang. Design and empirical evaluation of search software for legal professionals on the www. In *Information Processing and Management: an International Journal*, volume 36(2), pages 253–273. Elsevier Science, 2000.

[28] Caroline M. Eastman. 30,000 hits may be better than 300: precision anomalies in internet searches. In *Journal of the American Society for Information Science*, volume 53(11), pages 879–882, New York, USA, 2002. John Wiley & Sons, Inc.

[29] Stephen G. Eick, Joseph L. Steffen, and Eric E. Sumner Jr. Seesoft-a tool for visualizing line oriented software statistics. In *IEEE Transactions on Software Engineering*, volume 18(11), pages 957–968, Piscataway, NJ, USA, 1992. IEEE Press.

[30] David Ellis. A behavioural approach to information retrieval system design. In *Journal of Documentation*, volume 45(3), pages 171–212, London, UK, 1989. Aslib, The Association for Information Management.

[31] David Ellis and Merete Haugan. Modelling the information seeking patterns of engineers and research scientists in an industrial environment. In *Journal of Documentation*, volume 53, pages 384–403. Emerald Group Publishing Limited, 1997.

[32] FireFox. Mozilla's firefox web browser. (website), August 2006. http://www.mozilla.com/firefox/.

[33] Amy Fowler. Java heavy and light weight components. online article, August 2006. http://java.sun.com/products/jfc/tsc/articles/mixing/.

[34] George W. Furnas. The fisheye view: a new look at structured files. In *Readings in information visualization: using vision to think*, pages 312–330, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[35] George W. Furnas and Samuel J. Rauch. Considerations for information environments and the navique workspace. In *Proceedings of International Conference on Digital Libraries (DL 98)*, pages 79–88. ACM press, 1998. http://www.citeseer.ist.psu.edu/furnas98considerations.html.

[36] Google. Web search engine. (website), August 2006. http://www.google.com.

[37] Matthew Gray. Wandex indexing service. (website), 1993. http://en.wikipedia.org/wiki/Search_engine.

[38] Ratvinder S. Grewal, Mike Jackson, and Jon Wallis. Using visualisation to interpret search engine results. In *Proceedings of The Active Web 1999*, pages 15–25. (online proceedings), January 1999. http://seed.scit.wlv.ac.uk/papers/activeweb99.html.

[39] Grokker. Web search engine. (website), August 2006. http://www.grokker.com/.

[40] Masum Z. Hasan, Alberto O. Mendelzon, and Dimitra Vista. Visual web surfing with Hy+. In *CASCON '95: Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research*, page 28. IBM Press, 1995. http://www.citeseer.ist.psu.edu/hasan95visual.html.

[41] Masum Z. Hasan, Alberto O. Mendelzon, and Dimitra Vista. Applying database Visualization to the World Wide Web. In *ACM SIGMOD record*, volume 25(4), pages 45–49. ACM Press, 1996. http://www.citeseer.ist.psu.edu/hasan96applying.html.

[42] Susan Havre, Elizabeth Hetzler, Ken Perrine, Elizabeth Jurrus, and Nancy Miller. Interactive visualization of multiple query results. In *INFOVIS '01: Proceedings of the IEEE Symposium on Information Visualization 2001*, pages 105–112, Washington, DC, USA, 2001. IEEE Computer Society.

[43] Marti A. Hearst. *User interfaces and visualizations. In Baeza-Yates, Ricardo A.; Ribeiro-Neto Berthiers(Eds.): Modern Information retrieval*, chapter 10, pages 257–324. Addison Wesley Longman, 1999.

[44] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 76–84, Zurich, CH, 1996. ACM press. http://www.citeseer.ist.psu.edu/hearst96reexamining.html.

[45] Marti A. Hearst and Jan O. Pedersen. Visualizing information retrieval results: a demonstration of the tilebar interface. In *CHI '96: Conference Companion on Human Factors in Computing Systems*, pages 394–395, New York, USA, 1996. ACM Press.

[46] David G. Hendry and David J. Harper. An informal information-seeking environment. In *Journal of the American Society for Information Science*, volume 48(11), pages 1036–1048, New York, USA, 1997. John Wiley & Sons, Inc.

[47] Ron R. Hightower, Laura T. Ring, Jonathan Helfman, Benjamin B. Bederson, and James D. Hollan. Graphical multiscale web histories: A study of padprints. In *UK Conference on Hypertext*, pages 58–65. ACM Press, 1998. http://www.citeseer.ist.psu.edu/hightower98graphical.html.

[48] H. P. Hogeweg-de Haart. Characteristics of social science information: A selective review of the literature. part ii. In *Social Science Information Studies*, volume 4, pages 15–30, 1984.

[49] Lars Erik Holmquist. Flip zooming: Focus+context visualization of linearly ordered discrete visual structures. In *Breaking the Screen Barrier (thesis)*. Göteborg University, Dept. of Informatics, Göteborg University (publisher), May 2000. http://www.viktoria.se/fal/publications/play/2000/dissertations/leh/.

[50] Christoph Holscher and Gerhard Strube. Web search behavior of internet experts and newbies. In *Proceedings of the 9th International World Wide Web conference on Computer*

Networks : The International Journal of Computer and Telecommunications Networking, pages 337–346, Amsterdam, The Netherlands, 2000. North-Holland Publishing Co.

[51] Timo Honkela, Samuel Kaski, Krista Lagus, and Teuvo Kohonen. WEBSOM—self-organizing maps of document collections. In *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, pages 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997. http://www.citeseer.ist.psu.edu/honkela97websom.html.

[52] Adobe Inc. Photoshop tool : OS X version. (website), 08 2006. http://www.adobe.com/products/photoshop/.

[53] Peter Ingwersen. Cognitive perspectives of information retrieval interaction: Elements of a cognitive IR theory. In *Journal of Documentation*, volume 52, pages 3–50. Emerald Group Publishing Limited, 1996.

[54] Bernard J. Jansen and Udo Pooch. A review of web searching studies and a framework for future research. In Donald H. Kraft, editor, *Journal of the American Society for Information Science and Technology*, volume 52(3), pages 235–246. ASIS, John Wiley and Sons Inc, 2000.

[55] Bernard J. Jansen and Amanda Spink. An analysis of web searching by european alltheweb.com users. In *Information Processing and Management*, volume 41(2), pages 361–381, Tarrytown, NY, USA, 2005. Elsevier Science.

[56] Bernard J. Jansen and Amanda Spink. How are we searching the world wide web? A comparison of nine search engine transaction logs. In *Information Processing Management: An International Journal*, volume 42(1), pages 248–263. Elsevier Science, 2006.

[57] Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: a study of user queries on the web. In *ACM SIGIR Forum*, volume 32(1), pages 5–17, New York, USA, 1998. ACM Press.

[58] Bernard J. Jansen, Amanda Spink, and Jan Pedersen. A temporal comparison of altavista web searching: Research articles. In *Journal of the American Society for Information Science*, volume 56(6), pages 559–570, New York, USA, 2005. John Wiley & Sons, Inc.

[59] Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. In *Information Processing and Management*, volume 36(2), pages 207–227, Tarrytown, NY, USA, 2000. Elsevier Science.

[60] Shaun Kaasten and Saul Greenberg. Designing an integrated bookmark / history system for web browsing. In *Proceedings of Workshop on History Keeping in Computer Applications, College Park*. University of Maryland, December 1999. http://www.citeseer.ist.psu.edu/kaasten99designing.html.

[61] Shaun Kaasten and Saul Greenberg. Integrating back, history and bookmarks in web browsers. In *CHI '01: Extended Abstracts on Human Factors in Computing Systems*, pages 379–380, New York, USA, 2001. ACM Press.

[62] David Kirsh. The intelligent use of space. In *Artificial Intelligence*, volume 73(1-2), pages 31–68, Essex, UK, 1995. Elsevier Science Publishers Ltd. http://icl-server.ucsd.edu/ kirsh/Articles/Space/AIJ1.html.

[63] Matthew Koll. Major trends and issues in the information industry. (online article), 1999. http://www.asidic.org/news/techsumf99.html.

[64] Carol Collier Kuhlthau. Developing a model of the library search process: Cognitive and affective aspects. In *Reference Quarterly (winter)*, volume 28(2), pages 232–242. American Library Association, 1988.

[65] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. In *Intelligence*, volume 11(1), pages 32–39, New York, USA, 2000. ACM Press.

[66] Jan Lethen. Properties of pearson's correlation, November 1996.

[67] Ying K. Leung and Mark D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. In *ACM Transactions on Computer-Human Interaction (TOCHI)*, volume 1(2), pages 126–160, New York, USA, 1994. ACM Press.

[68] Maurice B. Line. Information requirements in the social sciences. In *Access to the Literature of the Social Sciences and Humanities*, pages 146–158, Queens College, City University of New York, New York, 1974. Queens College Press.

[69] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: detail and context smoothly integrated. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 173–176, New York, USA, 1991. ACM Press.

[70] Thomas M. Mann. Visualization of WWW-search results. In R.R. Wagner A.M. Tjoa, A. Cammelli, editor, *Proceedings of the International Workshop on Web-Based Information Visualization (WebVis'99) (in conjunction with DEXA'99, Tenth International Workshop on Database and Expert Systems Applications)*, pages 264–268, Florence, Italy, September 1999. IEEE Computer Society. http://www.citeseer.ist.psu.edu/mann99visualization.html.

[71] Thomas M. Mann. *Visualization of Search results from the World Wide Web*. Phd thesis, University of Konstanz, Germany, 2002.

[72] Thomas M. Mann and Harald Reiterer. Case study: A combined visualization approach for www-search results. In *Proceedings of the IEEE Visualization 1999 (Vis 99)*. IEEE, USA, 1999. http://www.citeseer.ist.psu.edu/492831.html.

[73] Gary Marchionini. *Information seeking in electronic environments*. Cambridge University Press, New York, USA, 1995.

[74] Gary Marchionini. Interfaces for end-user information seeking. In *Journal of the American Society for Information Science*, volume 43(2), pages 156–163. John Wiley and Sons, New York, January 1999. http://www3.interscience.wiley.com/cgi-bin/abstract/10049647/ABSTRACT.

[75] Mark McCahill, Farhad Anklesaria, Paul Lindner, Dan Torrey, and Bob Alberti. Gopher protocol. (website), 1991. http://en.wikipedia.org/wiki/Gopher_protocol.

[76] Ethical media. Opte project : A map of the internet. (website), august 2006. http://www.ethicalmedia.com/splash/mapofinternet/.

[77] Nancy Miller, Beth Hetzler, Grant Nakamura, and Paul Whitney. The need for metrics in visual information analysis. In *NPIV '97: Proceedings of the 1997 Workshop on New Paradigms in Information Visualization and Manipulation*, pages 24–28, New York, USA, 1997. ACM Press.

[78] Alan L. Montgomery and Christos Faloutsos. Identifying web browsing trends and patterns. In *Computer*, volume 34(7), pages 94–95, Los Alamitos, CA, USA, 2001. IEEE Computer Society Press.

[79] Nielsen's net ratings. Nielsen inc. (website), August 2006. http://searchenginewatch.com/showPage.html?page=2156451.

[80] Cartia news maps. Newsmaps: Topographic mapping of information. (website), 1995. http://mundi.net/maps/maps_015/.

[81] Jakob Nielsen. Usability engineering at a discount. In *Proceedings of the third international conference on human-computer interaction on Designing and using human-computer interfaces and knowledge based systems (2nd ed.)*, pages 394–401, New York, NY, USA, 1989. Elsevier Science Inc.

[82] Jakob Nielsen. Finding usability problems through heuristic evaluation. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380, New York, USA, 1992. ACM Press.

[83] Chris North and Ben Shneiderman. Snap-together visualization: A user interface for coodinating visualizations via relational schemata. In *Advanced Visual Interfaces*, pages 128–135. World Scientific Publishers, 2000. http://www.citeseer.ist.psu.edu/article/north00snaptogether.html.

[84] Chris North and Ben Shneiderman. Snap-together visualization: Can users construct and operate coordinated visualizations? In *Advanced Visual Interfaces*, pages 128–135. World Scientific Publishers, 2000. http://www.citeseer.ist.psu.edu/north00snaptogether.html.

[85] Lucy Terry Nowell, Robert K. France, Deborah Hix, Lenwood S. Heath, and Edward A. Fox. Visualizing search results: some alternatives to query-document similarity. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 67–75, New York, USA, 1996. ACM Press.

[86] Vicki L. O'Day and Robin Jeffries. Orienteering in an information landscape: how information seekers get from here to there. In *CHI '93: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 438–445, New York, USA, 1993. ACM Press.

[87] Robert N. Oddy. Information retrieval through man-machine dialogue. In *Journal of Documentation*, volume 33(1), pages 1–14. Emerald Group Publishing Limited, 1977.

[88] Christopher Olston and Ed H. Chi. Scent Trails: integrating browsing and searching on the web. In *ACM Transactions on Computer-Human Interaction*, volume 10(3), pages 177–197, New York, USA, 2003. ACM Press.

[89] Tiger OSX 1.4 operating system. Apple computers inc. (website), August 2006. www.apple.com.

[90] Windows O/S. Microsoft corporation. (website), August 2006. www.microsoft.com.

[91] Ramana Rao and Stuart K. Card. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 318–322, New York, USA, 1994. ACM Press.

[92] Harald Reiterer, Gabriela Tullius, and Thomas M. Mann. Insyder: a content-based visual-information-seeking system for the web. In *International Journal on Digital Libraries*, volume 5(1), pages 25–41. Springer-Verlag, 2005.

[93] Jonathan Roberts, Nadia Boukhelifa, and Peter Rodgers. Multiform Glyph Based Search Result Visualization. In *Proceedings of Information Visualization 2002*, pages 549–554. IVS, IEEE, July 2002. http://www.cs.kent.ac.uk/pubs/2002/1371.

[94] Jonathan C. Roberts, Nadia Boukhelifa, and Peter Rodgers. Visual Depictions of Search Results: using glyphs and coordinated multiple-views. In *YLEM Journal (Artists using science and technology)*, volume 24(2), pages 8–10. YLEM, CA, USA, January 2004. http://www.cs.kent.ac.uk/pubs/2004/2025.

[95] Jonathan C. Roberts and Edward Suvanaphen. Visual bracketing for web search result visualization. In *IV '03: Proceedings of the Seventh International Conference on Information Visualization*, page 264, Washington, DC, USA, 2003. IEEE Computer Society.

[96] George G. Robertson and Jock D. Mackinlay. The document lens. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 101–108, New York, USA, 1993. ACM Press.

[97] Tefko Saracevic. The stratified model of information retrieval interaction: Extension and applications. In *Proceedings of the ASIS (American Society for Information Science) Annual Meeting*, volume 34, pages 313–27. Wiley periodicals inc, 1997.

[98] Opera search engine. Opera software. (website), August 2006. http://www.opera.org.

[99] Marc M. Sebrechts, John Cugini, Sharon J. Laskowski, Joanna Vasilakis, and Michael S. Miller. Visualization of search results: A comparative evaluation of text, 2d, and 3d interfaces. In *Research and Development in Information Retrieval*, pages 3–10. ACM press, 1999. http://www.citeseer.ist.psu.edu/sebrechts99visualization.html.

[100] Ben Shneiderman. *Designing the user interface*. Addison-Wesley, 1998.

[101] Thumb shots. Thumbnails search engine. (website), August 2006. www.thumbshots.org.

[102] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a Very Large Altavista Query Log. In *ACM SIGIR Forum*, volume 33(1), pages 6–12. ACM press, 1998. http://www.citeseer.ist.psu.edu/silverstein98analysis.html.

[103] Amanda Spink. Study of interactive feedback during mediated information retrieval. In *Journal of the American Society for Information Science (JASIS)*, volume 48(5), pages 382–394. John Wiley and Sons, New York, 1997.

[104] Amanda Spink, Minsoo Park, Bernard J. Jansen, and Jan Pedersen. Multitasking during web search sessions. In *Information Processing and Management*, volume 42(1), pages 264–275. Elsevier Science, 2006.

[105] Anselm Spoerri. Metacrystal: visual interface for meta searching. In *CHI '04: Extended Abstracts on Human Factors in Computing Systems*, pages 1558–1559, New York, USA, 2004. ACM Press.

[106] Anselm Spoerri. Toward enabling users to visually evaluate the effectiveness of different queries or engines. In *Journal of Web Engineering*, volume 3(3-4), pages 298–313. Rinton Press, 2004. http://www.scils.rutgers.edu/ aspoerri/Publications/JWE2004public.pdf.

[107] Stephen K. Stoan. Research and library skills: An analysis and interpretation. In *College and Research Libraries*, volume 45(2), pages 99–109. Association of College and Research Libraries, 1984.

[108] Sue Stone. Humanities scholars: Information needs and uses. In *Journal of Documentation*, volume 38, pages 292–312. Emerald Group Publishing Limited, 1982.

[109] Bongwon Suh, Allison Woodruff, Ruth Rosenholtz, and Alyssa Glass. Popout prism: adding perceptual principles to overview+detail document interfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 251–258, New York, USA, 2002. ACM Press.

[110] Edward Suvanaphen and Jonathan C. Roberts. Explicit versus Implicit: An Analysis of a Multiple Search Result Visualization. In Ebad Banissi et al, editor, *Information Visualization*, pages 731–736. IEEE Computer Society, July 2004. http://www.cs.kent.ac.uk/pubs/2004/1991.

[111] Edward Suvanaphen and Jonathan C. Roberts. Textual Difference Visualization of Multiple Search Results utilizing Detail in Context. In Paul G. Lever, editor, *Theory and Practice of Computer Graphics*, pages 2–8, Bournemouth, June 2004. EGUK, IEEE Computer Society. http://www.cs.kent.ac.uk/pubs/2004/1927.

[112] R. Swan, J. Allan, and D. Byrd. Evaluating a Visual Information Retrieval Interface: AspInquery at TREC-6. In *CIIR technical report (Position paper for CHI 1998 Workshop on Information Exploration)*, 1998. http://www.citeseer.ist.psu.edu/swan98evaluating.html.

[113] Russell C. Swan and James Allan. Aspect windows, 3-d visualizations, and indirect comparisons of information retrieval systems. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181, New York, USA, 1998. ACM Press.

[114] Linda Tauscher and Saul Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. In *International Journal of Human-Computer Studies*, volume 47(1), pages 97–137, Duluth, MN, USA, 1997. Academic Press, Inc.

[115] Anne M. Treisman and Gary Gelade. A feature-integration theory of attention. In *Cognitive Psychology*, volume 12(1), pages 97–136. Elsevier Science, January 1980. http://www.ncbi.nlm.nih.gov/entrez/.

[116] Aravindan Veerasamy and Nicholas J. Belkin. Evaluation of a tool for visualization of information retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 85–92. ACM press, 1996. http://www.citeseer.ist.psu.edu/veerasamy96evaluation.html.

[117] Aravindan Veerasamy and Russell Heikes. Effectiveness of a graphical display of retrieval results. In *Proceedings of the 20th Annual International Conference on Research and Development in Information Retrieval*, pages 236–245, New York, USA, 1997. ACM Press.

[118] Java Programming Language (version 1.5). Sun microsystems. (website), August 2006. http://java.sun.com.

[119] W. Wertheimer. Laws of organization in perceptual forms (untersuchungen zur lehre von der gestalt ii). In *Psycologische Forschung*, volume 4, pages 301–350. Springer-Verlag, 1923.

[120] Wikipedia. online encyclopedia (website). http://en.wikipedia.org/, 2000.

[121] T. D. Wilson. On user studies and information needs. In *Journal of Documentation*, volume 37(3-15). Emerald Group Publishing Limited, 1981.

[122] T. D. Wilson. Models in information behaviour research. In *Journal of Documentation*, volume 55(3), pages 249–270. Emerald Group Publishing Limited, 1999. http://informationr.net/tdw/publ/papers/1999JDoc.html.

[123] Windiff. Microsoft sdk tools. (website), August 2006. http://en.wikipedia.org/wiki/WinDiff.

[124] Jeremy M. Wolfe. Guided search 2.0. a revised model of visual search. In *Psychonomic bulletin and review*, volume 1(2). Psychonomic Society Publications, 1994.

[125] Dietmar Wolfram, Amanda Spink, Bernard J. Jansen, and Tefko Saracevic. Vox populi: The public searching of the web. In *Journal of the American Society of Information Science*, volume 52(12), pages 1073–1074, 2001. http://www.citeseer.ist.psu.edu/wolfram01vox.html.

[126] Paint Shop Pro X. Corel Inc. (website), August 2006. http://www.corel.com/servlet/.

[127] Yahoo! Web search API, August 2006. http://developer.yahoo.com/search/.