Fetch: A Personalised Information Retrieval Tool

Innes Martin and Joemon M Jose
Department of Computing Science, University of Glasgow
17 Lilybank Gardens, Glasgow G12 8QQ, Scotland
{innes, jj}@dcs.gla.ac.uk

Abstract

Due to both the size and growth of the internet, new tools are needed to assist with the finding and extraction of very specific resources relevant to a user's task. Previously, the definition of relevance has been related to the matching between documents and query terms but recently the emphasis is shifting towards a more personalised model based on the relevance of a particular resource for one specific user. In this paper, we introduce our system, *Fetch*, which adopts this concept within an information-seeking environment specifically designed to provide users with means to describe a long-term multifaceted information need. By taking advantage of the way in which users bundle together groups of documents representing a particular topic, query languages as we know them can be taken to a higher and more useful level of abstraction. The agent personalises the search experience by using this information to formulate queries with the aim of returning documents relevant to the user's information need. In this paper we report on both qualitative and quantitative aspects of system use based on information collected in the pilot evaluation.

1. INTRODUCTION

Information seekers frequently rely on the Internet, and in particular web search engines, as their starting place, and in many cases the only place, from which to gain a better understanding of a topic. Formulating a good query is the first step in this process. It is widely acknowledged that query formulation, the transformation of a user's information need into a list of query terms, is a difficult task [16]. Users don't find this process of serialising their thoughts to be intuitive, often leading to poor queries and a widening of the gap between the actual need and stated request. The problem can be circumvented once users learn how documents are indexed and also have a basic idea of how the retrieval engine judges relevance. Unfortunately, as not all users study in the field of information retrieval, this knowledge is not something we can take for granted.

This problem is magnified when the information need is extremely vague ("*I don't know what I'm looking for, but I'll know it when I find it"*). This common scenario typically results in the formulation of short queries, consisting of 1 - 3 non-discriminatory terms. Inevitably, this leads to the connected problem of information overload where millions of documents are returned for a broad query. Certain users may trawl through pages and pages of results but in reality most will view the first few result pages before accepting these to be the best possible results, albeit mistakenly in most cases. This typically leads to extended or unproductive search sessions where any initial time constraints are waived and users become dissatisfied. In addition, they unknowingly miss relevant documents.

During the search process, a user's information need is constantly developing and so therefore is the query or queries associated with the search session. As users think, digest, interpret and problem solve, this information need can be influenced by any number of factors, for example, the quality of search results or the contents of a particular document. This dynamism has been exaggerated by what Robertson et al call the interactive revolution [13] highlighting the fact that IR systems have become increasingly interactive. Consequently, it is important to provide effective interfaces to facilitate this interactive information seeking process.

However, a major problem associated with information retrieval interfaces is the consistent lack of secondary notation - functionality to organise retrieved documents within the searching environment [9]. When sending queries to a search engine, the results are typically delivered via a simple static list providing no secondary notation. Bookmarking does give us the required functionality to save and arrange documents whilst searching however we lose flexibility due to the strict hierarchical structure enforced. Informality opens up an interface, giving access to a wider range of information seeking strategies and allowing searchers to be as structured as they wish. Hendry goes even further by claiming "layout matters – even novices, given the opportunity, seem to invent policies for laying out their search work".

In this paper an information seeking environment is proposed in which the information needs of the user can be detected. By observing the user's interaction with the system we can offset the lack of effective query formulation by allowing the user to implicitly describe a problem regardless of their level of understanding. So, instead of being constrained by time and accepting the first set of results, the user's interaction with this set of results can be used to formulate new queries with a view to satisfying a long-term search goal. Based on this evolved query, new information can be fetched pro-actively and presented to the user.

The remainder of the paper is organised as follows. Section 2 compares and contrasts a number of existing systems which aim to solve the problems discussed above. In Section 3 our system is introduced and then a pilot evaluation of the system is described in section 4. Finally, the main findings are summarised in section 5 along with a brief discussion on future work.

2. BACKGROUND / RELATED WORK

The difficulty users find in firstly describing an information need formally and also effectively organising relevant search results has spawned an active research area. We can address this first shortfall by moving the burden of query formulation away from the user towards a Personal Information Agent (PIA). The PIA monitors the information seeking environment and in doing so builds a model of the user's interests and preferences over time. Now the user need only formulate a vague query for each topic of interest and by monitoring the user's actions, the PIA can iteratively formulate new, more targeted queries. Current systems in this field not only exhibit this important functionality but also have specifically designed interfaces in order to facilitate these features.

Balabanovic et al [1] introduce one such agent which suggests a set of documents the user may find interesting. Users then explicitly evaluate each document before the system adjusts its parameters accordingly to try and improve performance. However, as details need to be extracted from all documents without the help of any external search engines, the system supports browsing but not searching and is therefore unsuitable when used within a large repository such as the Internet. Unfortunately, the interface is created using HTML forms and therefore rates poorly with respect to secondary notation.

WebMate [4] is a similar agent but can support both Internet searching and browsing. Using multiple vectors to keep track of user's interests, relevant documents can be suggested to the user. The system automatically attempts to learn the user's categories of interest by requiring the explicit marking of pages during normal browsing. However, this form of relevance feedback increases the cognitive load of the user which can cause inconvenience or introduce confusion. Also, by attempting to automatically learn the user's categories of interest, the system becomes susceptible to the inevitable problem of noise. The interface, once again, is qlet down by a lack of secondary notation.

The problem of information overload, often occurring with broadly scoped searches, can adversely affect a user's problem solving ability. Baldonado introduces a simple but powerful interface, SenseMaker [2], allowing users to develop strategies for coping with the loss of context occurring when a variety of independent sources are bundled together. SenseMaker helps users to recover a degree of context by giving them tools for iteratively organising citations and articles into higher-level bundles. This interface level aggregation facility enables users to view a collection from a variety of perspectives. Also, by increasing the fluidity between browsing and searching, the amount of exploration undertaken by a user rises. This occurs because real world tasks are time limited and therefore if the cost of switching between searching and browsing was reduced, users would be more likely to undertake deeper exploration.

As discussed in section 1, the functionality to organise results effectively is essential in any information seeking environment. Hendry presents SketchTrieve [9] which combines a graphic editor with dataflow notation in order to emphasise secondary notation. Searchers create search artefacts by laying out services on the display and connecting them together. Data from request-services flow into retrieval-services before results are computed and displayed. The user interface for SketchTrieve doesn't prescribe the structure for information displays or the ordering of dialogs too stringently. This under-determined structure promotes flexibility and helps searchers better understand their problem, just as pseudo code helps programmers. However, the display can become cluttered requiring extensive reorganisation.

Fetch adopts the flexible environment of SketchTrieve whilst incorporating a bundling technique allowing users to develop strategies for coping with the loss of context occurring when a variety of independent sources are viewed together. Over a period of time, through the observation of this bundling, the agent can build an accurate profile of the multifaceted information need and eventually recommend relevant documents without the need for users to explicitly mark documents.

3. FETCH : A PERSONALISED INFORMATION RETRIEVAL TOOL.

3.1 Introduction

In this section, we introduce our system. Before detailing the specific issues associated with *Fetch*, an introduction to the general concepts of the system is provided together with an insight on how it would typically be used. In the following text, numbers appearing in parenthesis provide reference to the components of the interface as seen in figure 3.1.

A query is executed via the search area (1) and results are returned together with a query-biased summary (2) for each link (3) in the result set. Links can then be dragged onto the workspace (4) and grouped together with similar documents to form bundles (5) analogous to the way in which related documents are placed in the same folder on a desktop. Bundles on the workspace are also represented in the overview panel (6) in order to complement the flexibility of the workspace with a more structured view. The agent will at some unspecified future point in time analyse the bundles belonging to each user and formulate a new query for each. The system notifies the user of this new information by changing the colour of the bundle on the workspace from green to red (7). By double clicking the updated bundle, instead of opening the bundle, a new search will be initiated using the associated query with results returned as before. Relevant links can then be dragged into new or existing bundles in the same fashion as before. The list of query terms can also be edited in the query editor (8) based on the quality of the first result set. Iterations of this form continue as long as the contents of the bundle are updated and thus the user's changing information need can be captured. The agent also checks for updated links on the workspace, alerting the user by changing the colour of the link icon from green to red.

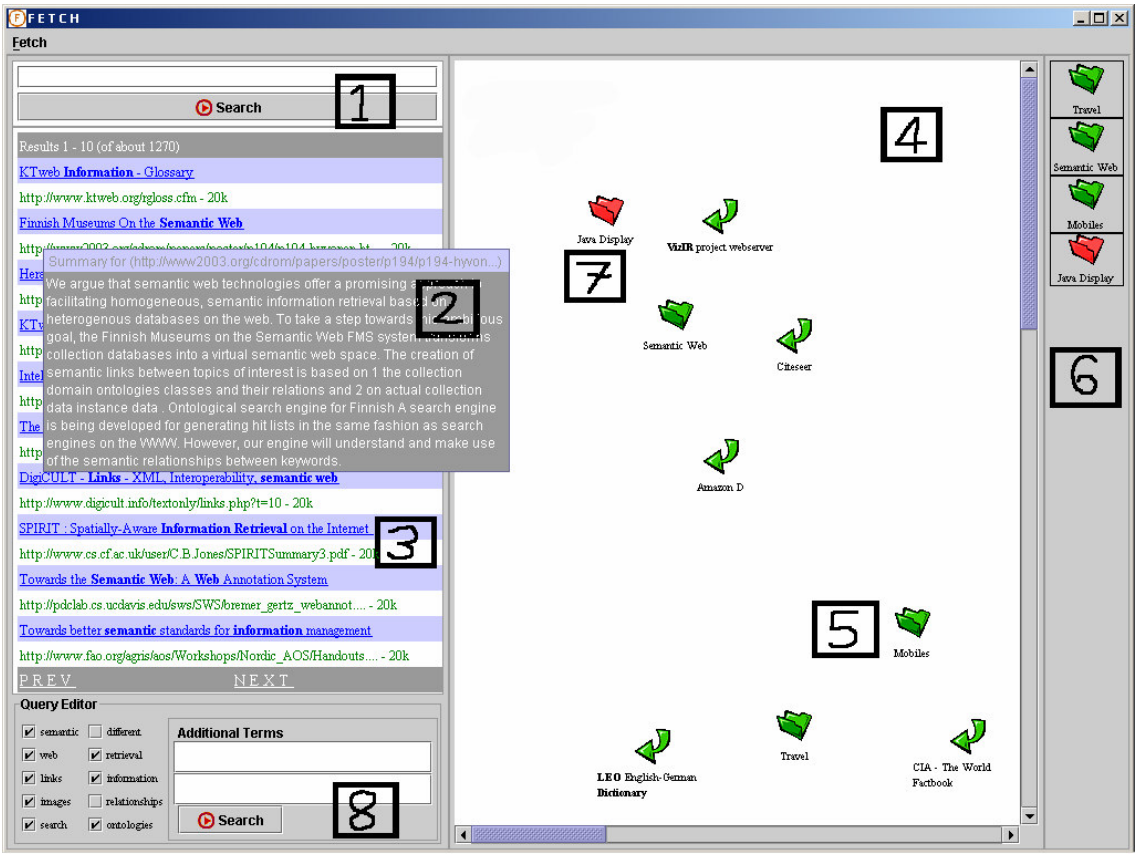


Figure 3.1: Main Screenshot

3.2 High Level Architecture

The high-level architecture of *Fetch* is based on the standard three-tier model and is shown in figure 3.2. The personal server mediates access to both the database and the search module in the usual way so as to decouple the two tiers. This architecture is also beneficial due to the ease in which both interaction and search activity can be monitored and logged. As the system may be expanded in the future to log browsing as well as searching, Internet access within the client is also forced to use the personal server as a proxy. The personal server is a Java servlet running on a Tomcat server; the personal server communicates with a Microsoft SQL Server 2000 database using JDBC; and the client/interface is implemented using Java Swing.

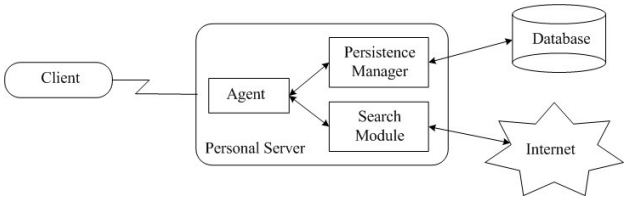


Figure 3.2: High Level Architecture

3.3 Data Model

The underlying data model is shown in figure 3.3 was designed with simplicity and extensibility in mind. Each user of the system is represented by a *User* object consisting of a collection of *FetchComponents* together with a *RelationTree*. This tree keeps track of the relations between components in the collection using tuples of the form <id1, id2> where component with id2 is the child of component with id1. For example, a *Link* belonging to a *Bundle* which in turn belongs to the *Workspace* would be represented using 2 such tuples. The root of the tree is the user's workspace, branch nodes represent bundles and all other *FetchComponents* can act as leaf nodes of the tree. By using this technique, the components themselves do not reference each other and thus any operations on them result only in the tree being altered. With respect to bundling, this proves to be a powerful solution allowing an arbitrarily large level of abstraction. For example, it is possible to create bundles within bundles within bundles while still using this reasonably trivial data model.

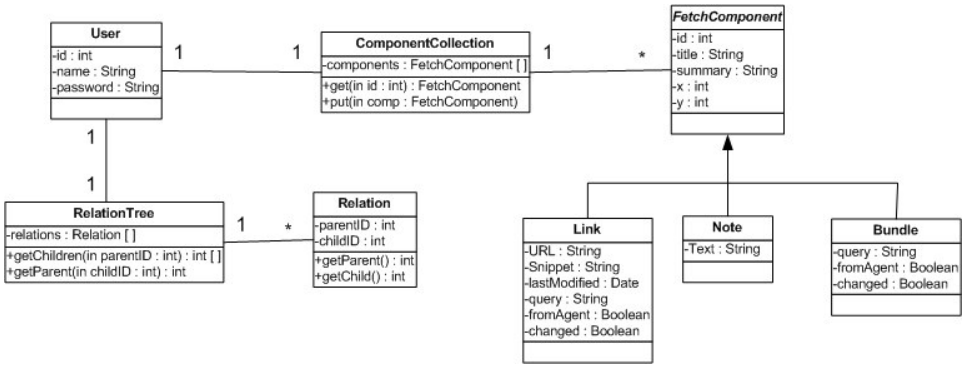


Figure 3.3: Data Model

3.4 The Client

The *Fetch* client, as seen in figure 3.1, uses a flexible interface comprising of a search area, workspace, web browser, and query editor, enabling users to search and organise results effectively. By providing users with an interface specifically designed to aid searching, we can take advantage of the valuable information contained within user actions. Although this feedback has to be described as explicit, the system is non-intrusive with users only interacting as and when necessary. Due to the design of the system, we can reap the benefits of noiseless explicit feedback without demanding anything from the user other than (s)he is using the system's intuitive bundling technique. Each component of the interface is discussed in detail below.

3.4.1 Search Area

With searching central to this application, a dedicated area of screen real estate is used to query repositories and display the results along with their summaries. The aim of this component is to provide a simple and familiar searching interface which can be complemented by other components within the application. Therefore, in order to minimise the cognitive load of the user, each results page has similar but simplified content to that produced by Google – one of the most popular internet search engines. The following information is displayed on each results page: total number of results; ranking range of the results we are viewing; title of each result (with search terms highlighted); URL and page size of each result. This page is dynamically created at query time and shown within a simple Java Swing component capable of displaying HTML. By clicking the hyperlink title of a particular result, the web-page is displayed in a new window using the in-built browser

detailed later in this section. To help users judge the relevance of each result, query-biased summaries are also returned from the personal server, appearing in a pop-up box when the user hovers the mouse over a result. This prevents users from having to visit the page or rely on the two-line, often irrelevant abstract as their only assessment of its relevance [15]. Finally, relevant hyperlinks can be dragged and dropped onto the workspace as described previously.

3.4.2 Workspace

The workspace is designed to be a large, flexible area used to aid problem solving through the adoption of the bundling technique introduced in [8]. A bundle is defined as "a grouping of information collected, selected, elaborated and structured during problem solving". In field observations, bundles appear to be a widely used means of managing information to support diverse, complex, often simultaneous tasks [8]. With respect to *Fetch*, users can organise web-links on the workspace into bundles and attach notes to these bundles in the same way, for example, a post-it note would be stuck to a pile of papers residing on a desk. This method enables users to make use of both space and abstraction to logically break down their long-term information need and thus provides traditional problem solving techniques within our information-seeking environment. Users can either drag links from the search area or create links on the workspace and these can be repositioned using the conventional drag and drop technique. To create a bundle, users can right click the workspace and choose the 'create bundle' option from the resulting pop-up menu. Links and notes can then be dragged into the bundle in a similar way to which files are added to a folder in a standard WIMP environment. All components on the workspace can be opened by way of double clicking and are displayed in their respective viewers i.e. links are shown in the in-built browser; bundles in the bundle viewer and notes in the note viewer. In order to prevent users from frequently having to open up these viewers, useful summaries are shown in a pop-up box when users hover the mouse over a component on the workspace. The pop-up for a link is the query-biased summary, for a bundle it is a hierarchical representation of its contents and for a note it is the body of the note.

3.4.3. Overview Panel

Through the use of scrollbars, the large size of the workspace provides an obvious scalability issue. As the number of components increases, users can easily lose track of where bundles exist on the workspace and require means to regain their orientation. It is important to provide the user with context by complementing the flexibility of the workspace with a more structured view. The overview window does just this with each bundle on the workspace being represented in the window by a smaller version of its actual icon. Workspace components can also be dragged and dropped onto these icons as a way of easily adding items to bundles without first locating their positions on the workspace. By double clicking on an icon in the overview window, the scrollbars are adjusted accordingly to position the corresponding bundle at the centre of the viewable workspace. In the case where several bundles are located closely together on the workspace, an arrow also briefly points to the bundle in question to remove any ambiguity.

3.4.4 Query Editor

Although using a query-generation algorithm in the agent which exhibits a low noise level is important, there is some evidence that allowing end-users to interact directly with feedback improved not only actual (measured) retrieval performance but also perceived performance, trust in the system, and subjective usability [12]. Seo et al [14] describes the agent as looking over the shoulder of the user but in this case the user is also looking over the shoulder of the agent. By allowing users to edit queries formulated by the agent we can make sure that all remaining noise is instantly eliminated. Based on the quality of a previous result set, the

query editor can also be used to produce more effective and targeted queries by adding/deleting terms as appropriate. The query editor component contains the list of query terms, each with a checkbox, enabling users to quickly remove terms as necessary. Query terms can be added using the text fields supplied.

3.4.5 Browser

An important requirement when designing an information-seeking environment is the ability for users to view documents returned during the search process in a way they are accustomed. With respect to web-pages, users expect browsers with bookmarking, history and caching functionality, capable of handling javascript, applets, flash and pdf files as well as standard HTML. Rather than *externally* spawning a new browser window and thus losing all the feedback that browsing provides, it was decided to keep this functionality within the information-seeking environment. By using a Java browser that runs on native code, it is possible to provide the speed and functionality users expect without losing the ability to monitor interaction. This method also allowed application-specific components to be added to the browser, for example, a button to add the current web-page to an existing bundle.

3.5 Agent

The client interacts with the agent to access persistence as well as the ability to search and browse the Internet. Based on the way in which users break down their information need by bundling documents together, the agent suggests new documents which may be of interest as well as monitoring existing documents on the workspace. A new query is attached to every bundle stored in the database which has been updated since the previous agent iteration. These new queries are formed by extracting relevant information from each constituent element. For a link, the query-biased summary provided by the search component and for a note, the entire text is used. The number of terms used in this new query is dynamic in order to prevent information overload, for example, the agent will formulate a query which never returns more than 65,000 documents by altering the number of query terms until this requirement is satisfied.

When studying information seeking behaviour on the Web, Choo et al [5] discovered that 58% of pages visited during each session were re-visits. The agent eliminates the need to frequently check these web-pages for updates, as the user is alerted when web-pages referred to by links on the workspace, have been updated. This is achieved by storing the last modified time stamp for each link in the database. During each cycle, the current last modified time stamps are retrieved using HTTP connections and a flag in the data model is set for each updated page. The client renders these updated links on the workspace using a different coloured icon in order to alert the user to the new information.

Although the granularity of the agent's cycle can be set to instantaneous (i.e. the agent starts suggesting new documents as soon as the workspace has been updated), the purpose of the system was to satisfy long-term searching goals and consequently the agent executes one cycle every 24 hours. This produces a similar effect to the personalised newspapers presented in [4] [7] where content matching the users' interests was delivered each morning.

4. EVALUATION

Traditional evaluation schemes for information retrieval tools do not consider the interaction between user and system in an information seeking situation [11]. Recent trends are moving away from the typically system-centred evaluation towards a more user-based evaluation. However, Borlund [3] states that increasing realism and moving away from a system-based evaluation approach can introduce problems where we have to relinquish control over

experimental variables, observability and repeatability. To combat this, simulated work task situations can be used as an experimental sub-component and by giving each user the same task, we can introduce control and compare search results across systems or users. A simulated work task situation is defined by Borlund as "a semantically rather open description of a scenario" and can be used to trigger and develop a simulated information need by allowing for subjective user interpretations of the situation.

4.1 Experimental Methodology

8 test subjects were used for this pilot evaluation: 5 students currently studying at the University of Glasgow and 3 non-students from various professions. Subjects were given the evaluation instructions detailing that they should use the system whenever it suited them, at least once every day, for 10 days. By allowing free access to the system, work could be completed at any time and consequently the usual pressures inflicted upon subjects by observation were eliminated. Subjects were given 3 simulated work task situations and instructed to choose 2 on which to base their simulated long-term information need. In addition to this, subjects were encouraged to use the system to monitor a topic in which they had a genuine personal interest. All important interactions (search statistics, bundling etc.) were automatically logged in the database by adding a simple component to the Personal Server designed to capture all relevant requests and responses. A post-evaluation questionnaire was also used to gauge sentiment toward the more qualitative properties of the system such as usability, interface design etc.

4.2 Hypotheses

Our main hypothesis is that *Fetch* is an effective tool for satisfying long-term information need. This hypothesis is split into two sub hypotheses in order to provide a connection to the empirical data recorded:

1. *Fetch* is an effective tool for satisfying long-term information need.
   1.1 *Fetch* can be used to model the dynamic nature of a long-term multifaceted information need.
   1.2 *Fetch* helps overcome the problems of information overload and query formulation.

4.3 Results and Analysis of Log Data

4.3.1 Hypothesis 1.1 – *Fetch* can be used to model the dynamic nature of a long-term multifaceted information need.

We investigate two factors relating to hypothesis 1.1 below:

- The content of each bundle and associated query evolves over time.

Through tracking each bundle from creation time, we can build up a picture of how each develops over time. Looking firstly at the contents of bundles, as expected, the number of links per bundle grows over time (figure 4.1). However, the differing rates of growth observed suggest 3 distinct stages during the lifetime of a bundle. Stage 1 is the loading stage where subjects identify a facet of their information need, create a bundle and add some links relevant to that topic. This is followed by stage 2 where the bundle grows at a steadily decreasing rate and in some cases a third stage where the bundle is no longer updated. This final stage occurs due to one of the following reasons: the bundle represents a shorter-term information need which has been satisfied and is now therefore redundant; the bundle suitably represents a long-term information need and is now used solely to check for updated versions of the web-pages contained within; or the user has lost interest in the topic represented by the

bundle. The growth of links throughout the system also showed similar characteristics to the bookmaking behaviour reported by Cockburn et al [6] where the rate of bookmark addition heavily outweighed the rate of deletion. During the evaluation, the total rate of link creation compared to link deletion was 361 to 38.

Due to the coupling between a bundle's contents and its associated query, as bundles evolve over time, so too do these queries. In figure 4.2, we track the query associated with each bundle over its lifetime and show the average percentage of new terms for each day based on the previous days query. As the age of the bundle increases, it can be seen that the associated query starts to converge with less and less new terms being suggested by the agent. This can be explained in terms of the slowdown in change of a bundle's content highlighted in the previous graph.

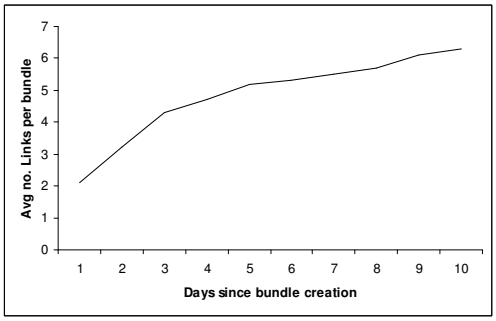


Figure 4.1: Bundle Contents

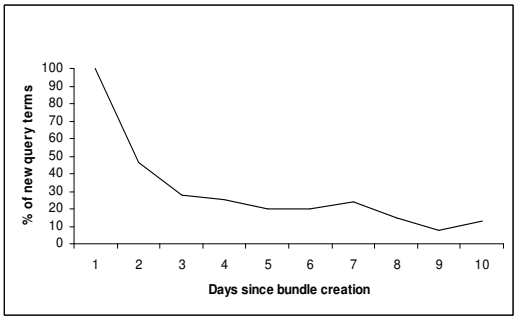


Figure 4.2: Bundle Queries

- The quantity of bundles for each user increases with time as new facets of the information need are established

By logging each time a bundle is created or deleted, we can investigate how subjects used the system to divide up their information need and organise documents returned during search sessions. All users adopted the bundling technique with each finishing the evaluation with on average 6.8 bundles. The minimum and maximum number of bundles was 3 and 16 respectively with the former accounting for two users who simply used one bundle per evaluation task. 4 subjects created a bundle hierarchy to separate their different information need facets into sub-facets with the remaining 4 subjects creating all bundles on the top-level workspace. Of the bundle hierarchies that were created, none were extended beyond the 2nd level.

As one could expect, the rate of bundle creation was greater during the early stages of the evaluation with subjects using this setup phase to lay out all facets of their initial information need. As the evaluation continued, the number of bundles grew steadily as new facets of the information need were discovered or existing facets were subdivided for further development. This can be seen in figure 4.3 below.

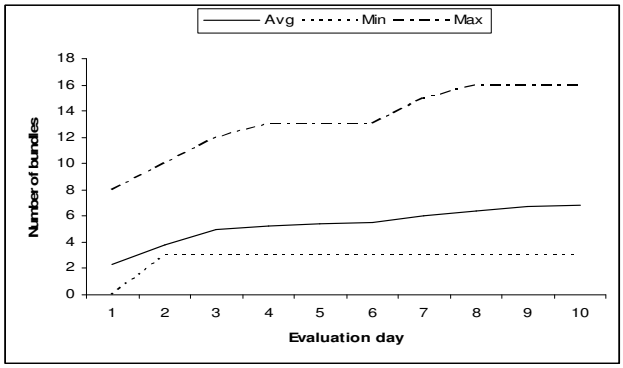


Figure 4.3: Number of Bundles

4.3.2 Hypothesis 1.2 – *Fetch* helps overcome the problems of information overload and query formulation.

We investigate two factors relating to hypothesis 1.2 below:

- The number of documents returned from each agent-assisted search is sensible

Firstly, looking at the average number of terms in each user-formulated query (figure 4.4), we can see that searchers only use an average of 2.6 terms which is consistent with results published in [10], a study of search behaviour by Jansen et al. Short queries are one of the main causes of information overload and inevitably the average number of documents returned for user-formulated queries was a rather overwhelming 342,000 (see figure 4.5). When considering queries formulated by the agent, the average number of query terms increases to 8.2, with the number of documents retrieved falling to approximately 3900. This is undoubtedly a more realistic number of documents for users to process whilst still high enough to retrieve a significant number of relevant results. However, in the worst case, the agent produces over-targeted queries retrieving only a handful of documents which, despite the possibility of high precision, is unacceptable in terms of recall. When editing queries suggested by the agent, users submitted 5.3 terms per query returning an average of 34,000 documents. The minimum and maximum number of documents returned using this method of query formulation was 1,300 and 141,000 respectively. This suggests that the either the agent recommends queries containing too many terms or some of the terms are too discriminatory or both. This view is supported when we look at the statistics from hypothesis 2.2 below.

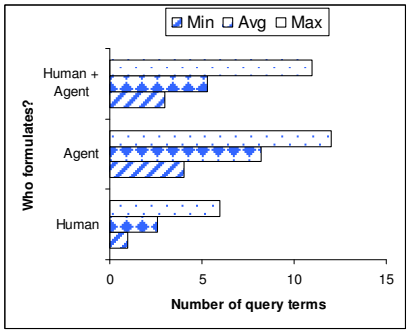


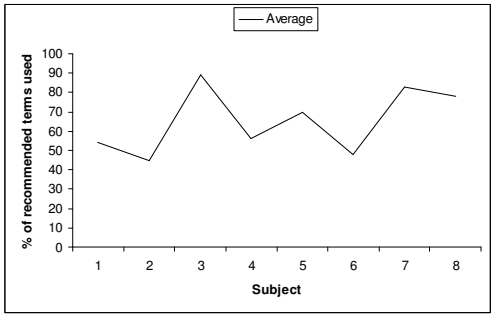
Figure 4.4: Query Size

	Human	Agent	Human + Agent
Min	17,300	2	1,300
Avg	342,000	3,900	34,300
Max	2,570,000	64,500	141,000

Figure 4.5: Documents Returned

- The queries formulated by the agent are effective

When an agent formulated query is sent to the search module, based on the quality and/or quantity of the result set, users can edit the initial query. Terms can be added and/or deleted before the reformulated query is submitted. This process can deliver a good indication as to the relevance of the originally recommended terms. Firstly, let us consider the average number of recommended terms used (i.e. not deleted from the list) in the reformulated query when the agent formulated query is edited. As seen in figure 4.6, no users kept less than 45% of the recommended terms for the reformulated query, with an overall average of 65%. Secondly, if we consider the number of recommended terms used as a percentage of all terms in the reformulated query, the results are even more significant. Figure 4.7 shows that recommended terms consistently make up more that 74% of the total query terms used and on average count for 91.0%. This suggests that users are satisfied with selecting from the list of recommended terms and so don't add their own and/or they struggle to think of additional terms even with some indicative examples.



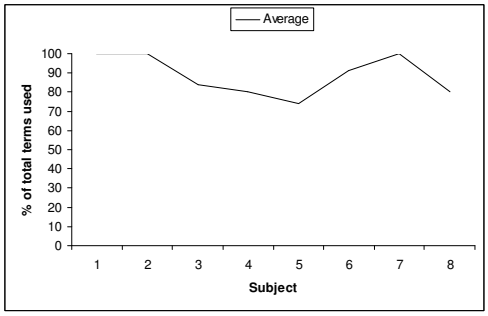


Figure 4.6: Recommended terms used **Figure 4.7: Recommended terms in new query**

The claim above ascertains to what degree users perceive query terms to be relevant, an important factor in any information retrieval system, but let us now concentrate on actual relevance i.e. the quality of result sets between query modes. Our definition of relevance, while typically ambiguous in web retrieval, is in this case trivial due to the nature of the interface. A document is said to be relevant if it is dragged from the search results and placed anywhere on the workspace. Admittedly, this is a strict definition of relevance but it is used solely to compare the quality of result sets between query modes. As seen in figure 4.8, 6 out of 8 subjects found that the agent formulated queries returned more relevant documents than conventional querying. Overall, 8.9% of non-agent results were deemed to be relevant, compared with 11.2% recommended by the agent. This relative improvement of 25.8% suggests the documents returned by the agent formulated queries were substantially better in terms of relevance. However, the most successful results occurred when queries suggested by the agent were user-edited. Using this query mode, 14.1% of documents were judged to be relevant, a 58.4% improvement over conventional querying. These results show that the documents returned by agent-formulated queries are significantly more relevant than those returned when queries are formulated by the user. However, our results suggest that the most effective query mode is a combination of human and agent formulation where users simply eliminate noise from the query-generation algorithm. Users, despite their apparent inability to formulate queries from scratch, can easily form effective queries by eliminating inappropriate query terms when presented with a list.

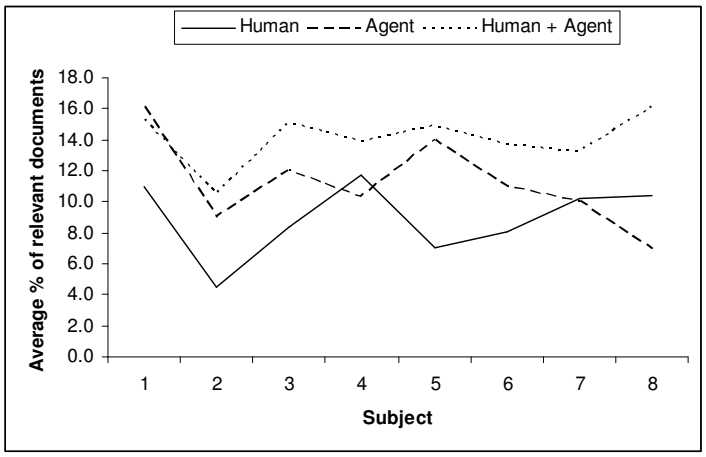


Figure 4.8: Relevant Documents

4.4 Post-evaluation Questionnaire

After using the system for 10 days, users were asked to complete a questionnaire inviting them to indicate, on 7-point semantic differentials (as seen is figure 4.9), their opinions regarding the evaluation task and usability of the system. With respect to usability, the differentials focused on the searching experience, the agent and also general interface issues. The arrangement of positive and negative descriptors was randomised to encourage users to think carefully prior to answering apparently similar questions. In order to gain further insight into the users' preferences, participants were also asked in open questions to specify which features of the system they liked/disliked and were given the chance to elaborate on responses given to the semantic differentials.

The task we asked you to perform was..?
1. (clear -> unclear) **2**. (simple -> complex) **3**. (familiar -> unfamiliar) **4**. (interesting -> boring) **5**. (pleasant -> unpleasant)
Searching using Fetch was..?
1. (fast -> slow) **2**. (easy -> difficult) **3**. (familiar -> unfamiliar) **4**. (pleasant -> unpleasant) **5**. (satisfying -> frustrating)
The documents returned by the personal agent were..?
1. (relevant -> irrelevant) **2**. (important -> unimportant) **3**. (useful -> useless) **4**. (appropriate -> inappropriate) **5**. (complete -> incomplete)
The bundling technique was..?
1. (useful -> not useful) **2**. (intuitive -> confusing) **3**. (easy to use -> hard to use) **4**. (familiar -> unfamiliar)
When interacting with the system, I felt..?
1. (in control -> lost) **2**. (comfortable -> uncomfortable) **3**. (confident -> unconfident)
The system you have used to complete this task is..?
1. (effective -> ineffective) **2**. (satisfying -> frustrating) **3**. (reliable -> unreliable) **4**. (flexible -> rigid) **5**. (useful -> useless) **6**. (novel -> standard) **7**. (fast -> slow)

Figure 4.9: Semantic Differentials

The resulting 8 sets of scores were averaged for each question (see figure 4.10) and will now be discussed along with the responses given to the open questions.

The evaluation was generally well received with subjects perceiving the task to be clear (2.2) and reasonably pleasant (2.9). Borlund [3] states that a simulated work task situation can only replace a genuine information need if the user can relate to the task given. In our carefully chosen tasks, users found they could engage themselves in the problems and did not find the tasks to be boring (2.6).

The aim of the searching component was to provide the ability to search multiple sources from within the Fetch environment whilst maintaining a familiar and accepted interface. This aim was accomplished with users rating searching as easy to use (2.2), satisfying (2.5) and very familiar (1.9). Also, despite the extra time inflicted by creating summaries for each URL in the results pages, users were still satisfied with the speed in which the searches were performed (2.1).

The agent was an integral part of the system and so the quality of documents suggested to the user was important to the overall acceptance of the system. During the evaluation, users found the documents returned to be relevant (2.3) and useful (2.6). However, in the open questions, two subjects raised concern as to why only a handful of documents were returned by the agent on several occasions. This is concurrent with findings in section 4.3.2, again suggesting that the agent can occasionally formulate over targeted queries, either with respect to the number of query terms used or the over-discriminatory nature of these terms.

The usability of the bundling technique was well rated with subjects finding this method of organisation both useful (1.9) and easy to use (1.7). With respect to intuitiveness (2.6), three subjects couldn't quite understand why they "had to actually create a bundle" before using it. Instead of having to place components into an existing bundle, this group of users would simply rather position components geographically close together on the workspace without actually creating a physical bundle entity. Despite these grievances, at a semantic level the bundling technique was seen as a real strength of the system with users reporting it both "useful to be able to group related links together" and "good to organise result without leaving the searching environment".

When interacting with the system, most subjects felt reasonably confident (2.8) and in control (3.1) and at all times, however, the application was scored poorly on flexibility (4.5). Subjects felt the interface was over-defined with all components visible throughout regardless of whether or not they were currently in use.

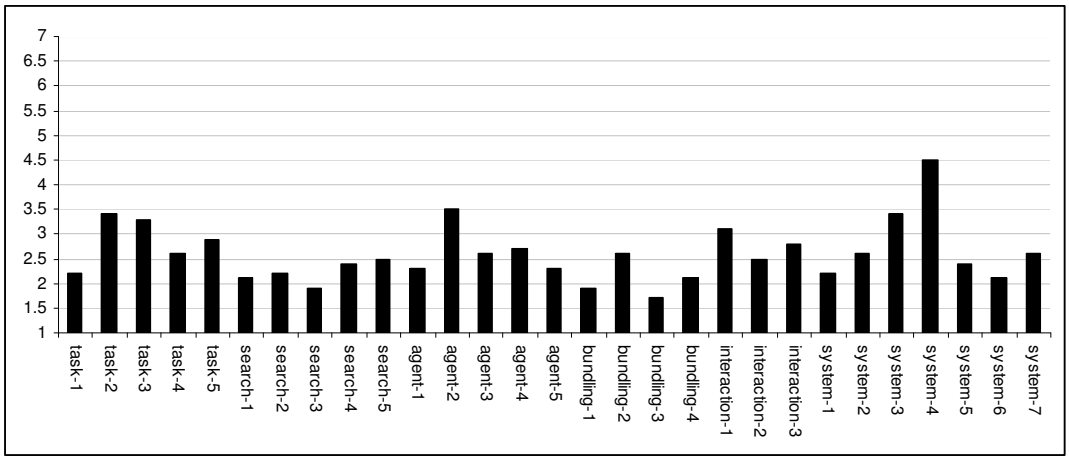


Figure 4.10: Semantic differential means (value range 1-7, lower=better)

5. SUMMARY / FUTURE WORK

We have designed, deployed and evaluated a system that aims to alleviate the problems of query formulation, information overload and lack of secondary notation by providing an effective problem solving environment used to express a long-term information need. By simply formulating vague queries and organising their information objects using spatial arrangement, users can take advantage of the targeted queries provided by the agent. As time progresses, these queries evolve in sync with the user's information need, becoming increasingly targeted as a more detailed model of the user develops. Initial findings from the pilot evaluation are positive. Users successfully adopted the bundling technique and during our evaluation the number of bundles grew steadily as new facets of the information need were discovered or existing facets were subdivided for further development. The contents of these bundles evolved with time, allowing the agent to formulate increasingly targeted queries to assist users during search sessions. On average, queries provided by the agent successfully decreased the problem of information overload but occasionally these queries only returned a handful of documents. Users found documents returned by the agent to be more relevant than those returned via conventional querying with even more significant results when the agent-formulated queries were edited. The problem of query formulation is lessened in this way by allowing users to formulate short queries consisting of non-discriminatory terms and then stating "I want documents like these". In a similar manner, when the agent-formulated queries are displayed, users can simply say "I don't want those query terms" which helps eliminate noise and return relevant documents without the burden of improving query formulation skills. Future evaluation will involve a larger and more varied sample using a slightly updated version of the interface based on feedback received during this pilot evaluation.

6. ACKNOWLEDGEMENTS

The work reported in this paper is funded by the Engineering and Physical Research Council (EPSRC), UK grant number GR/R74642/01.

7. REFERENCES

[1] Balabanovic, M., & Shoham, Y. (1995) *Learning Information Retrieval Agents: Experiments with Automated Web Browsing.* AAAI-95 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments.

[2] Baldonado, M., & Winograd, T. (1997*) Sensemaker: An information exploration interface supporting the contextual evolution of a user's interests*, Proceedings of the ACM Conference on Human Factors in Computing Systems.

[3] Borlund, P. (2000) *Experimental Components for the Evaluation of Interactive Information Retrieval Systems*, In: Journal of Documentation, Vol. 56, no. 1, 2000, pp71-90.

[4] Chen, L., & Syraca, K. (1998) *Webmate: A Personal Agent for Browsing and Searching*, Proceedings of the 2nd International Conference on Autonomous Agents, pp132-139.

[5] Choo, C.W., Detlor, B., & Turnbull, D. (1999) *Information Seeking on the Web: An Integrated Model of Browsing and Searching*, Proceedings of the Annual Conference of the American Society for Information Science, pp.127-135.

[6] Cockburn, A., & McKenzie, B. (2001) *What Do Web Users Do*? *An Empirical Analysis of Web Use*, International Journal of Human-Computer Studies (in press).

[7] Croft, W.B., Cronen-Townsend, S., & Lavrenko, V. (2001) *Relevance Feedback and Personalization: A Language Modelling Perspective*, DELOS Workshop: Personalization and Recommender Systems in Digital Libraries.

[8] Delcambre, L., Maier, D., Bowers, S., Deng, L., Weaver, M., Gorman, P., Ash, J., Lavelle, M., & Lyman, J. (2001) *Bundles in Captivity: An Application of Superimposed Information*, Proceedings of the 17th International Conference on Data Engineering.
[9] Hendry, D. G., (1996) *Extensible Information-Seeking environment*, PhD thesis, Robert Gordon University, Aberdeen.
[10] Jansen, B. J., Spink, A. and Saracevic, T. (2000) *Real life, real users and real needs: A study and analysis of users queries on the Web*, Information Processing and Management, 36(2), 207-227.
[11] Jose, J. M., Furner, J. F., and Harper, D. J. (1998). *Spatial Querying for Image Retrieval: A User-Oriented Evaluation*. In Croft, B., Moffat, A., Van Rijsbergen, C. J., Wilkinson, R., & Zobel, J. (Eds.), Proceedings of the Twenty First ACM SIGIR Conference on Research and Development in information retrieval, pages. 232-240. ACM Press.
[12] Koenemann, J., & Belkin, N.J. (1996) *A case for interaction: A study of Interactive Information Retrieval behaviour and effectiveness,* Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, ACM, pp 205-212.
[13] Robertson, S. E. & Hancock-Beaulieu, M. M., (1992) *On the evaluation of IR systems*, Information Processing and Management, 28 (4), 457-466.
[14] Seo, Y.W., & Zhang, B.T. (2000) *Learning user's preferences by analyzing web browsing behaviours*, Proceedings of Int'l Conference on Autonomous Agents '2000, pp. 381-387.
[15] White, R. W., Jose, J. M. and Ruthven, I. (2003). *A task-oriented study on the influencing effects of query-biased summarisation in web searching*. Information Processing & Management, 39(5):707-733.
[16] Cool, C., Park, S., Belkin, N.J., Koenemann, J. & Ng, K.B. (1996) *Information seeking behaviour in new searching environment,* CoLIS 2, pp403-416.