Abstract

The Internet as an information resource is exploding in size, depth and complexity. In an age where human attention is worth so much, it is becoming increasingly time consuming to decipher the relevant information from the irrelevant and accelerate the laborious process of web browsing. Imagine delegating these tasks to your own, trained, personal Internet assistant, who looks over your shoulder, giving you suggestions and shortcuts to the web pages you, personally find relevant? The developments in intelligent agent technology beckon for application to the challenging Internet domain.

This report documents the investigation and implementation of a system of smart web browsing tools, designed to reach beyond the experimental stage, to a fully functional and effective end-user application.
Acknowledgements

I would like to thank my supervisor, Chris Hogger, for his constructive yet witty thoughts during our project meetings, and his willingness to spend extended periods of his time during the crucial points of the project.

I would also like to thank Frank Kriwaczek for his support and helpful advice, not only during the timeframe of the project, but throughout my degree.

Finally, I would like to thank my family and friends for their support, patience and resourcefulness during the project.
# Contents

1 **Introduction** .............................................................................................................. 1  
  1.1 Overview ........................................................................................................... 1  
  1.2 Objectives ......................................................................................................... 2  
  1.3 Report Structure .............................................................................................. 2  

2 **Background to Smart Web Browsing** .................................................................. 4  
  2.1 Intelligent Agents ............................................................................................ 4  
     2.1.1 What is an Agent? .................................................................................. 4  
     2.1.2 What Makes an Agent Intelligent? ......................................................... 5  
     2.1.3 Types of Intelligent Web Agents ........................................................... 5  
  2.2 Web Mining ..................................................................................................... 5  
  2.3 Existing Approaches to Smart Web Browsing ....................................................... 7  
     2.3.1 Web Mining Techniques ........................................................................ 7  
     2.3.2 Existing Smart Web Browsing Tools ...................................................... 9  
  2.4 Critique of Existing Approaches ...................................................................... 12  
     2.4.1 Tools to Employ ................................................................................... 12  
     2.4.2 System Architecture ............................................................................. 12  
     2.4.3 Web Mining Techniques ....................................................................... 14  
     2.4.4 Implementation Languages ................................................................... 16  
     2.4.5 Usability ............................................................................................. 17  
  2.5 Summary ....................................................................................................... 18  

3 **Decisions and Technical Challenges** .................................................................... 19  
  3.1 Set of Tools .................................................................................................... 19  
     3.1.1 Personal Shortcut Recommendations .................................................... 19  
     3.1.2 New Document Recommendations ...................................................... 19  
     3.1.3 Bookmark Recommendation .................................................................. 20  
  3.2 Web Mining Techniques .................................................................................. 20  
     3.2.1 Autonomy ........................................................................................... 20  
     3.2.2 Document Representation ..................................................................... 20  
     3.2.3 User Interest Modelling ......................................................................... 21  
     3.2.4 Access Pattern Modelling ..................................................................... 25  
  3.3 System Architecture ....................................................................................... 31  
     3.3.1 Real-time Usage ................................................................................... 31  
     3.3.2 Information Collection .......................................................................... 31  
     3.3.3 Distributed Architecture ......................................................................... 31  
     3.3.4 Implementation Language ..................................................................... 32  
     3.3.5 Future Expansion Modularity ............................................................... 33  
  3.4 Usability .......................................................................................................... 33
D.7  getNumberOfDocuments ................................................................. 95
D.8  getProfileID ................................................................................... 96
D.9  logurl ............................................................................................. 96
D.10 readAccessPattern ................................................................. 97
D.11 readProfile .................................................................................. 97
D.12 removeBookmark .......................................................................... 97
D.13 saveAccessPattern .................................................................. 98
D.14 saveProfile .................................................................................. 98
D.15 updateDF .................................................................................... 98
D.16 updateTDF ................................................................................... 98

E  Toolkit API ......................................................................................... 100
Table of Figures

Figure 1: Estimation of the number of hosts on the internet [ISC04] ......................... 1
Figure 2: Web mining in relation to other forms of data mining and retrieval [WEBMIN] .... 6
Figure 3: Taxonomy of Web Mining [Coo97] ............................................................... 6
Figure 4: Example of a three dimensional vector space with three terms and three
documents ............................................................................................................... 7
Figure 5: Example of a web server log ........................................................................ 13
Figure 6: WebMate proxy server architectural approach to tracking web usage [Che98] .. 13
Figure 7: Example access pattern to model Nanopoulos limitations .............................. 15
Figure 8: Summary of .NET and Java architectures [INFORMIT] ............................... 16
Figure 9: Strategic features of .NET and Java framework [INFORMIT] ....................... 17
Figure 10: How multiple documents would be represented in a category collection of
similar documents in the Vector Space Model ...................................................... 21
Figure 11: How an irrelevant document would be represented with regard to a category in
the Vector Space Model .................................................................................... 21
Figure 12: New proposal for multiple category user interest profiling handling outliers ... 23
Figure 13: User interest profiling algorithm pseudo-code ........................................... 25
Figure 14: Diagram illustrating the new proposal to determine real-time, sequential,
frequent itemsets ................................................................................................ 27
Figure 15: Pseudo-code for real-time sequential frequent itemset generation algorithm .. 28
Figure 16: Pseudo-code for the generation of sequential frequent itemsets ...................... 29
Figure 17: Example of sequential frequent pattern generation ..................................... 30
Figure 18: Conceptual system component interaction overview ................................... 37
Figure 19: Insight into the design of the proxy server application ............................... 39
Figure 20: Insight into the design of the intelligence server application ....................... 39
Figure 21: Insight into the design of the client application .......................................... 40
Figure 22: Designed graphical user interface for the client application ......................... 41
Figure 23: Unified Modelling Language diagram of a user interest profile abstract data type
.................................................................................................................... 43
Figure 24: Example of a two-term document user interest profile in XML form ............ 44
Figure 25: Example of a DMSMessage as an XML string ........................................... 45
Figure 26: Modified Mentalis proxy server console application ................................. 46
Figure 27: Implemented intelligence server console application ............................... 46
Figure 28: Learner class diagram ............................................................................. 46
Figure 29: ProfileManager flowchart implementation.................................................. 47
Figure 30: Implemented client end-user interface windows application ....................... 48
Figure 31: Recommendation popup message windows .............................................. 49
Figure 32: Security mechanism to protect user privacy. .......................................... 50
Figure 33: Database schema.................................................................................. 50
Figure 34: Example of a stored procedure in the database........................................ 51
Figure 35: Client application interface and its accordance to Nielsen’s 10 Usability Heuristics ........................................................................................................... 59
Figure 36: Breakdown of users who tested the software............................................. 61
1 Introduction

1.1 Overview

Availability of information is no longer an issue on the Internet. The dynamic nature of the evolving Internet and the growth of information content requiring extraction is becoming a more demanding task for users alone as each day progresses [Her96]. Users are faced with the daunting task of identifying the relevant information from an ever-growing sea of inadequate or inappropriate data made available on the Internet today. The escalating rate of new information sources is illustrated by the number of hosts on the web in Figure 1.

Whilst search engines provide some information to web users, their use is limited, due to the lack of providing structural information and categorisation, filtering or interpretation of the documents. We are witnessing a paradigm shift in human-computer interaction from direct manipulation of computer systems to indirect management [Mou97]. These factors call for intelligent tools in the form of web agents to work alongside the user to provide a higher-level understanding of a user-specific web space.

![Internet Domain Survey Host Count](Source: Internet Software Consortium (www.isc.org))

Figure 1: Estimation of the number of hosts on the internet [ISC04]

Everyone uses the internet as an information resource. However, each time a person browses the World Wide Web (WWW), there is no-one and nothing to remember who they are and what their habits are. It is clear that the process of information discovery could be greatly assisted if there was a way of making it personalised [Mou97]. However, after a user currently views a web page or queries a search engine, their interest in that topic will rarely finish at that point, and will persist for a given period of time. The most convincing reason to adopt a set of browsing tools is due to this characteristic of a user’s persistence of interest [Lie95].

Novice web users would feel more comfortable in an environment where subtle suggestions for user-specific pages of interest and recommendations of superior browsing
techniques were at hand. Instead, they are bewildered by an ever increasing wealth of information. Experienced web users can be found to visit a small repeat collection of web sites frequently, but each time they browse these sites, their identity is unrecognised and cannot benefit from any form of a user-specific recommendations. This is portrayed with around a one fifth of web users having difficulty in finding pages they’ve already visited [Bou01].

Utilising web mining techniques to combat the explosive growth of internet information shown in Figure 1, this project aims to automatically construct user-specific browsing models, generate intelligent inferences of these observed browsing activities and provide users with a set of tools that help them to browse the websites that they commonly visit in a more effective fashion. In order to achieve this, the inherent challenges in web mining must be addressed [Han00]:

- The complexity of web pages is far greater than that of any traditional text document collection.
- The web is a highly dynamic information source.
- Only a small portion of the information on the web is truly relevant or useful to each web user.
- The web seems to be too huge for effective data warehousing.
- The web serves a broad diversity of user communities.

1.2 Objectives

The objectives for this project are as follows:

- Research ideas and concepts of a system that could analyse user observed browsing activities automatically to underpin a set of tools that can help users to browse the websites they commonly visit in a more effective fashion.

- Decide on the design of a system of smart web browsing tools and the underlying series of techniques that will be used to facilitate them.

- Implement the system that meets the design criteria with sufficient reliability and quality that it can be released to, and used by, untrained web users.

- Test and evaluate the system by the effectiveness of the web mining techniques and the usability of the implemented system of tools.

1.3 Report Structure

There are five major sections to this report:

**Background to Smart Web Browsing (page 4)**

This section provides research into the concepts and techniques required for this project. A critique of the existing approaches is included, identifying the important aspects of a smart web browsing tool.
Decisions and Technical Challenges (page 19)
This section draws inferences from the background research by examining the strengths and weaknesses of the existing approaches. Decisions are made to address these technical challenges, with new approaches to the modelling of user interests and access patterns proposed.

Specification and Design (page 34)
These sections contain a formal specification for the implementation of a system of smart web browsing tools followed by a presentation of an overall design for each of the components of the system.

Implementation and Testing (page 43)
The final implementation of the system is described, with the problems faced during implementation discussed. Test results of the effectiveness and usability of the system are then critically evaluated in these sections.

Conclusion (page 62)
In this final section, the overall result of the project is compared to the original objectives and presents a set of thoughts and ideas for opportunities of future work.
2 Background to Smart Web Browsing

This section provides a brief insight into the concepts and techniques required for this project. An introduction to the intelligent agents research area is followed by a summarisation of the existing tools with a discussion of them.

2.1 Intelligent Agents

A tool that would prove to be useful to a user's browsing experience requires a degree of intelligence, performing delegated tasks on behalf of the user. A research area that coincides with this description is known as intelligent agents.

2.1.1 What is an Agent?

Agents are defined by Russell [Rus95] as software or hardware entities that perform some set of tasks on behalf of users with some degree of autonomy. Agents therefore employ some knowledge or representation of a user's interests and goals, when acting with a measure of independence whilst performing its operations. They are optimized to perform their operations, and in doing so are useful in reducing the information workload, which has become a fact of life for web users. An intelligent agent can be applied to search for patterns of interest by using learning and intelligence techniques in classification, clustering, summarisation and generalisation.

An agent can be split up into three components. A sensing element for detecting changes in the environment, a learner mechanism for dealing with the new event the environment, and an advisor for providing the deduced task solution to the user.

Agents can be classified into multiple categories based on their characteristics. An agent can display more than one of these characteristics including:

- Learning Agent: Adapts to the requirements of its user and automatically changes its behaviour due to environmental changes. Learning techniques abolish the need for programming rules, but still allow the agent to adapt to changes in the user’s environment.

- Autonomous Agent: Takes action without user intervention, making decisions of various levels of complexity on its own, operating concurrently, either while the user is idle or taking other actions.

- Interface Agent: Provide an easy-to-use interface to complex systems such as the Internet, but also to computers in general.

In the context of intelligent web agents, these characteristics would entail:

- Understanding a web user’s browsing habits and interests.
- Autonomously discovering documents.
- Providing assistance in the form of recommendations while browsing.
2.1.2 What Makes an Agent Intelligent?

In the context of agents, intelligence is portrayed by the ability to choose among various courses of action, plan, communicate, adapt to changes in the environment and learn from experience. These abilities give intelligent agents a set of characteristics that include independence, learning, inference, adaptability and creativity. These characteristics are required if tasks are to be delegated to an agent and performed in a manner which mirrors that of a personal assistant.

2.1.3 Types of Intelligent Web Agents

Intelligent web agents act on behalf users supporting tasks such as browsing, searching, collecting, comparing and filtering web data. These agents are categorised into one or more of the following categories depending on the tasks they assist/perform:

**Browsing Agents**

A specific branch of intelligent agents are known as browsing agents, essentially “looking over the user’s shoulder” whilst the user interfaces with their browser. The goal of a browsing agent is to provide the user with effective advice to help them locate the relevant information required from their browsing experience. The agent is an autonomous process, running in the background of the computer while the user is the one with absolute control over the browsing path.

**Search Assistants**

Search assistants are an evolving form of search engines, tracking the content of information resources and providing the user with a system that records user information to help achieve efficient content searching. The search assistant removes the need for the user to find a formal query to represent the topic of interest. Changes to relevant content are reported to the user by periodic communication.

**Matchmaking and Comparison Agents**

This type of agent provides the user with results of intelligent data mining on internet content. The agent monitors specific sites or resources and then applies heuristics to imply relations among the data. For example, such an agent would offer answers to the queries requesting “the cheapest known product” or “an expert in some subject”.

**Filtering Agents**

Filtering agents filter dynamic data to provide users with pre-managed incoming information. One example of this would be providing a newsgroup with filtered content dictated by a prearranged query. In other words, determining the relevant content and only allowing through the information deemed useful, based on a given user profile.

2.2 Web Mining

Web mining is defined by [Coo97] as the discovery and analysis of useful information from the WWW. Web mining is used to extract interesting and potentially useful patterns and implicit information from artefacts or activity related to the WWW. Web mining in relation to other forms of data mining and retrieval is illustrated using Figure 2. The diagram demonstrates the fact that web mining is performed on an unstructured source, i.e. web sites.
Web mining is categorised into three distinct categories, shown in Figure 3:

**Web Content Mining**
Web content mining is the automatic search of information resources available online [Coo97]. As a process, web content mining goes beyond keyword extraction since web documents present no machine-readable semantic. The two groups of web content mining approaches concentrate on different aspects. Agent based approach directly mines document contents. Database approach improves the search strategy of the search engine with regard to the database it uses.
**Web Structure Mining**

Whilst web content mining focuses on the internal structure of a web document, web structure mining tries to discover the link structure of the hyperlinks at the inter-document level.

**Web Usage Mining**

Web usage mining is defined as the discovery of user access patterns from web servers. Web servers record and accumulate user interaction data each time a user makes a request for resources. Analysing these web access logs can reveal patterns regarding a user’s browsing habits through the web server.

### 2.3 Existing Approaches to Smart Web Browsing

An understanding of the existing approaches to smart web browsing entails a close look at the available set of web mining techniques and some existing smart web browsing tools. An insight into how these existing tools applied web mining techniques and a summary of their findings provide for subsequent further discussion.

#### 2.3.1 Web Mining Techniques

**TF-IDF Vectors**

Documents can be represented as vectors under the vector space information retrieval paradigm [Sal83]. In this model, each document is represented by a vector of weights, each of which corresponds to a feature, a word in most cases as in Figure 4. The word ordering information in the document is usually not retained and hence the name “bag-of-words”. TF-IDF word vectors are created from each processed document by assigning each word in the document a weight.

![Figure 4: Example of a three dimensional vector space with three terms and three documents](image.png)

Weighting is applied to each document term based on the following formula:
Background to Smart Web Browsing

\[ W[\text{word}] = TF[\text{word}] \log \left( \frac{D}{DF[\text{word}]} \right) \]

Where TF[\text{word}] is the count (term frequency) of \text{word} in the document being processed, D is the total number of existing documents, and DF[\text{word}] is the number of documents in which \text{word} occurs at least once (document frequency).

A collection of TF-IDF vectors can be used to represent a web user’s interests.

**Cosine Similarity**

Cosine similarity can be used to determine the similarity between two TF-IDF vectors. Cosine similarity is achieved by computing the dot product of the respective TF-IDF vectors.

\[ \text{Sim}(V_jV_k) = \frac{V_j \cdot V_k}{|V_j| \times |V_k|} \]

Where \( V_j \) and \( V_k \) are the matched vectors.

**K-Nearest Neighbour**

Han [Han00] defines k-nearest neighbour as a classification method where classifiers are based on learning by analogy. Training samples are described by n-dimensional numeric attributes. Each sample represents a point in an n-dimensional space. In this way, all of the training samples are stored in an n-dimensional pattern space. When given an unknown sample, a k-nearest neighbour classifier searches the pattern space for the k training samples that are closest to the unknown sample. These k training samples are the k “nearest neighbours” of the unknown sample. Closeness is defined in terms of Euclidean distance, where the Euclidean distance between two points, \( X = (x_1, x_2... x_n) \) and \( Y = (y_1, y_2... y_n) \) is

\[ d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \]

**Neural Networks**

Neural networks are well suited for data mining tasks due to their ability to model complex, multi-dimensional data. As data availability has magnified, so has the dimensionality of problems to be solved, thus limiting many traditional techniques such as manual examination of the data and some statistical methods. Although neural networks are quite adept at finding the hidden patterns in data, they do not directly reveal their findings to the developer.

**Naïve Bayes**

The Bayesian classifier is a probabilistic method for classification whether a given vector represented page will be of interest to a user, based on a given threshold. It can be used to determine the probability that an example \( j \) belongs to class \( C_i \) given values of attributes of the example:
Background to Smart Web Browsing

\[ P(C_i \mid A_1 = V_1 \& \ldots \& A_n = V_n) \]

If the attribute values are independent, this probability is proportional to:

\[ P(C_i) \prod_k P(A_k = V_k \mid C_i) \]

Both \( P(A_k = V_k \mid C_i) \) and \( P(C_i) \) may be estimated from training data. To determine the most likely class of an example, the probability of each class is computed. An example is assigned to the class with the highest probability.

This algorithm computes a single value for a given document. The idea is that interesting documents should receive higher values whereas uninteresting documents should receive lower values. Therefore this method is used in combination with an interest threshold.

**Porter Stemmer**

One technique for improving retrieval effectiveness is stemming [Por80, Fra92]. Stemming provides a method of automatic conflation of related words, usually by reducing them to a common root form. An example is shown below:

“… as the four engineers followed their lead engineer across the pier, it became apparent that something was wrong with the engineered plan…”

In this passage, the occurrences of the terms engineer, engineers and engineered are each 1. This passage would be better represented if the stem “engineer” is counted instead of the full term.

The advantage of stemming is that common words will be grouped and represented under the same stem, providing a more statistically accurate estimation for the number of occurrences of a given word. The disadvantage of stemming is that information about the full terms will be lost or unrepresented.

**Sequential Association Rules**

Apriori is an influential algorithm for mining frequent itemsets for association rules [Agr94]. Apriori employs an iterative approach known as a level-wise search, where \( k \)-itemsets are used to explore \((k+1)\)-itemsets. The Apriori property employed in association rule mining can be applied to mining sequential pattern because if a sequential pattern of length \( k \) is infrequent, its superset (of length \( k+1 \)) cannot be frequent [Han00].

**2.3.2 Existing Smart Web Browsing Tools**

The following illustrates a brief insight into existing approaches to similar problems.

**Search Engines**

Search engines [GOOGLE, MSN] focus on indexing the WWW, providing a method of finding web documents based on a given query. A common problem with search engines is that they frequently return documents that are irrelevant to what the user was expecting. This can be due to two reasons. Firstly, the keywords used to form the query are a poor choice with regard to expected document set returned. A lack of experience is the primary
cause for this weakness, where users are unfamiliar with the interface and the options available when forming an accurate query. Secondly, the search engine itself provides a poor set of indexes for the collection of document. Even the perfect query can be limited by the accuracy of the set of indexes it determined and currently holds.

**WebWatcher**

WebWatcher [Arm97] is a goal-oriented browsing assistant that makes link suggestions by highlighting links that may lead to the pre-defined goal. WebWatcher requests an initial explicit goal from the user, and the e-mail address to keep track of the user’s interests. The agent tracks user behaviour and constructs new training examples from the encountered hyperlinks. Each hyperlink is evaluated using a utility function based on the Page, the Goal, the User and the Link. The function assigns a value from 0 to 1 to the links, suggesting the best ones.

The system uses TF-IDF vectors, cosine similarity, statistical prediction and Boolean functions.

**Letizia**

Letizia [Lie95, Lie97] is an autonomous interface agent that assists a user browsing the WWW. Letizia doesn’t require the user to provide an explicit goal like WebWatcher. Instead, it automates a browsing strategy consisting of a breadth-first search, within the boundary of linked pages, augmented by a set of content and context based heuristics to infer the goal. For example, when a user saves a bookmark for a document, this is interpreted as interest in the content of the document. Similarly, if a document link is not clicked, this is interpreted as disinterest in the subject.

Upon request, Letizia will make recommendations on which links on the current page the user should follow based on the interest profile inferred by the agent. The recommended links for the user to follow are presented in a preference ordered list.

There is little detail given by Lieberman as to how Letizia represents a user profile and the exact learning techniques employed. What is apparent is that Letizia treats documents as a set of keywords/topics.

Letizia is implemented in Macintosh Common Lisp with an adapted Netscape web browser.

**PowerScout**

PowerScout [Lie01] is another tool from Lieberman adopting a similar user interest profile as Letizia. PowerScout’s approach to recommendations differs from Letizia’s forward scouting by using search engines to determine new pages of interest that match a given user’s interests.

Like Letizia, PowerScout is implemented with an adapted Netscape web browser.

**WebMate**

WebMate [Che98] is an agent that helps users to effectively browse and search the web by monitoring user’s browsing behaviour based on a user selecting a page as “I like”. The profile constructed consists of a collection of TF-IDF document vectors under the Vector Space Model [Sal83] using cosine similarity function to determine vector similarity.
WebMate compiles a personal newspaper based on the user’s profile by either querying popular search engines or by automatically trawling a list of user-defined URLs.

**Lira**

Lira [Bal95] is an agent that autonomously searches the WWW for interesting web pages. Documents are represented as vectors under the Vector Space Model [Sal83]. Lira incorporates stemming [Por80] so that the terms in the document vectors are represented by their stems only and not whole words. Weighting of the document terms are performed using a sophisticated TF-IDF scheme which normalises for a given document length:

\[
ui = \frac{0.5 + 0.5 \frac{tf(i)}{tf_{\text{max}}} \left( \log \frac{n}{f(i)} \right)}{\sum_{j \in T} \left( 0.5 + 0.5 \frac{tf(j)}{tf_{\text{max}}} \left( \log \frac{n}{f(j)} \right)^2 \right)}
\]

Where \(ui\) is the weight of the term \(I\), \(n\) is the number of documents in dictionary, \(tf(i)\) the document frequency of the term \(I\), \(tf_{\text{max}}\) is the maximum term frequency over all words in document \(T\) and \(f(i)\) the document frequency of the term \(i\). Document relevance is computed by calculating the dot product \(V.M\) where \(V\) is the document vector and \(M\) is the user profile vector.

The user assigns presented pages with an evaluation \(e_i\) in the range \(-5\) to \(+5\) and the profile is updated by a variation of the classical relevance feedback [Roc71]:

\[
\hat{M} \leftarrow M + \sum_{i=1}^{n} e_i V_i
\]

**Almathaea**

Almathaea [Mou97] is a co-evolution model of information filtering agents that adapt to the various users’ interest and information discovery agents that monitor and adapt to the various on-line information sources. Moukas states that users should rely on search engines and use the result of their queries as starting points for their agents. Therefore, Almathaea dispatches several of these filtering agents to query search engines and filter responses using an adapted TF-IDF measure:

\[
W = H_c \cdot T_f \cdot \log \left( \frac{N}{df} \right)
\]

where \(H_c\) a header constant, \(T_f\) is the frequency of the keyword in the current document, \(N\) is the total number of documents that have already retrieved by the system and \(df\) the document frequency of that keyword.

**Footprints**

Footprints is a prototype system created to help people browse complex web sites by visualizing the paths taken by users who have been to the site before [Wex97]. The system comprises of two main pieces: a front end and a back end. The back end runs in a
batch mode at a given time (say once per night), processing web logs of a single web site. The front end reads the data file output by the backend and is used to create a customised site map for a user as they enter the new web site.

Footprints presents the results of statistical analysis performed on data representing the history of interaction between users and a given web site. It provides a variety of tools such as maps, path views, annotations comments or signposts by other users. These paths are shown as a graph of linked document nodes, with the links colour-coded to visualize the frequency of use of the different paths.

### 2.4 Critique of Existing Approaches

Existing approaches have contributed to furthering the research into intelligent web agents. Each approach is constructively criticised based on their set of tools employed, system architecture, web mining techniques, implementation languages and usability.

#### 2.4.1 Tools to Employ

To date, a plethora of tools have already been implemented in an attempt to help user browse the web. These tools provide assistance in the form of browsing, searching and filtering the information available to a web user. To date, no single approach has incorporated a combination of tools to address each of these three tasks of a web user. With each tool addressing a single task, web users are left with the prospect of running numerous tools concurrently (possibly alongside different browsers) to benefit from the varying types of advice. A brief description of the functionality of existing tools is given below.

**New Document Recommendation**
This type of tool employed by Letizia, PowerScout, WebWatcher, WebMate, Lira and Almathaea recommends documents that are new to the user based on their interest profile. This tool either traverses links from the user’s current position, queries search engines or generally searches the WWW to find recommendations.

**Popular Document Recommendation**
This type of tool makes recommendations of popular pages to be kept as bookmarks. Letizia bases recommendations on user interest similarity.

**Forward Browsing Path Recommendations**
This type of tool makes predictions as to popular browsing paths based on access patterns. In Footprints, these recommendations are based on other previous users’ patterns.

#### 2.4.2 System Architecture

Two clear approaches to understanding a web user’s behaviour have emerged from the research. The first approach uses data from logs of web users that have visited a particular web site. In this case, the information collected can be anything from one day old to many months. Whilst Footprints showed how this information could be used to help new users to
Background to Smart Web Browsing

a site, only statistical inferences could be drawn from this information. There was no identification of a user and therefore no persistence in providing user-specific feedback. Therefore, it appears that the best method for providing real-time personal assistance to a web user will involve learning from on-the-fly web logs and combining these with a user profile to create a persistence of interest.

The second approach to understanding behaviour is by tracking real-time web data from a user. Letizia and WebWatcher achieve this by modifying the user’s browser application. However, over the years, a major drawback to this approach has ultimately emerged with regard to browser usage. Letizia uses Netscape and WebWatcher uses Mosaic as their modified browsers. At the time of conception, these browsers were popular with web users. The change in web browser usage has resulted in a position now, where Netscape and Mosaic represent only 2% of the web browser population [W3S04]. Whilst it can be tempting to assume that Internet Explorer will maintain its 81.7% coverage [W3S04] and remain as the dominant browser in today’s environment, it can be seen that a web tool that is browser dependent would have a highly limited future and would require changes to a complex piece of software.

WebMate successfully demonstrated the proxy server approach, logging HTTP requests to track web usage. The proxy server approach opens up a client-server relationship between the user and the assistant. Whilst this provides the system with distributed architectural advantages, there is still a group of web users that do not look favourably upon a centralised web browsing agent due to the requirements of the user to surrender trust and personal information [Som99]. This issue of trust can be addressed by either providing security around the personal information or implementing a system that has the option of running on a single machine. No existing approach implements a system that has a degree of distributed flexibility.

Another factor that determines the appearance of a real-time agent’s dependency on the system architecture is when to incorporate the learning. Web users will always pause at some point in their browsing session, usually to read the document [Lie95]. Lieberman
used this time as a leverage, by overlapping evaluation and search during the "idle time". Therefore, in order to facilitate a real-time system, the architecture should take advantage of this time to compute any expensive calculations or deductions.

2.4.3 Web Mining Techniques

2.4.3.1 Web Content Mining
A user interest profile must be modelled for each individual user in order to perform web content mining on their behalf. Various techniques in learning a user interest profile have been demonstrated by existing tools.

WebMate [Che98] learns a user interest profile incrementally and continuously, the user profiling algorithm only runs when a user explicitly marks a document as “I like it”. This entails two weaknesses for the system. Firstly, if only the positive documents are selected for incorporation, the values of D and DF for these documents are approximated. Secondly, the user still has to provide some degree of subconscious thought and decision making when explicitly marking documents. This marking is also undertaken at the same time as browsing the web. Whilst an accurate profile may be deduced, there would be a significant amount of user concentration time lost during in this process. Decisions such as whether or not a page should be marked will need to be continuously addressed. These types of decisions are not generally welcomed by web users if they need to be addressed on a continuous, long term basis in order to provide accurate recommendations. Lieberman [Lie97] suggests that even the simplest of interactions that lasts just a couple of seconds can be disruptive enough to the user's workflow that he or she won't bother to do it. Therefore the effectiveness of a tool that has a degree of dependence on user interaction will ultimately lower its usability. Schwab [Sch02] too concludes that user interest should not rely very much on user ratings, but rather take passive, unobtrusive observations about users into account as far as possible. Also, the central source of information about the user is his or her web navigation behaviour, and especially the set of selected objects.

Existing approaches have modelled user interest either as a single global interest goal [Arm97, Bal95] or multiple categories of interest [Lie97, Lie95, Che98]. In the single global interest profiling technique, all interests determined from the documents viewed by a user are amalgamated into a single interest representation. A drawback of this approach is illustrated using the following example. If a user has an interest in “Indian food” and “ladies fashion” these topics would form an interest in “Indian food ladies fashion”. Therefore, a document that contains information on “fashion food” would be deemed interesting. This emphasises the need to model multiple distinct categories of interest.

A user may find a topic both interesting and important in the short term. However, this topic could end up falling from a user's attention in a short space of time. A user profile indicating their interests ought to incorporate a factor of decay [Lie95], due to these dynamic aspects of a user’s interests.

Another issue regarding user profiling is consistency. If a system is highly dependent on a user explicitly rating pages such as WebMate [Che98], then that system is susceptible to inconsistent marking due to human emotion. This is where a user decides to mark a document as “I like” even though they may not necessarily like that document. This provides the system with a degree of false information, which could subsequently reduce the accuracy of the recommendations.
2.4.3.2 Web Usage Mining Techniques

Whilst the main techniques used by existing approaches have relied heavily on clustering, the sequentiality characteristic of web usage has been rarely used. Sequentiality refers to the order of document requests of a web user, i.e. the order in which the user viewed the web pages.

Footprints [Wex99, Wex97] proved successful employment of sequential pattern mining on web logs to denote the most popular paths through a single web site of users as a tour guide for subsequent users (in the future). However, drawbacks with Footprints’ approach in the context of a personalised intelligent agent include:

Site specific
The web logs of a single specific site are mined and the user patterns on this site alone contribute to the advice determined by Footprints. Unless this technique was employed on every site on the WWW, then a user would not be able to benefit from the advice. Given that there currently exists almost 250,000,000 web hosts [ISC04], employing a piece of software on a significant number of sites would become an impossible task. A solution that was user-based and not site-based has the advantage of being functional to those who web users who want to use it.

No personal advice
The advice given by Footprints is a representation of the combined browsing habits of users previous to the site and not a representation of the personal habits of the web user presently accessing the site. While the advice given by this collection of users is extremely useful for the new users to the site as a tour guide, the advice is not user-specific. For example, whilst 99% of users view page X, when the a repeat user who is never interested in page X visits the site, they are still advised to go to page X regardless of their own personal habits.

Chen [Che98b] evaluates the idea of Maximal Forward references as a more accurate way of modelling user access patterns. He states that a path such as {ABCDCE} which includes backward traversals as well as forward traversals is not taken into consideration. Instead the example path given would be better represented as {ABCD, ABCE}, i.e. two paths.

Nanopoulos [Nan01] presents the problems of finding similar traversal patterns. However, the approach takes into consideration the structure of a graph (i.e. the links on a page). The solution proposed is therefore confined to determining patterns only from those pages that are linked. The drawback of this approach is that true web browsing habits are not modelled. An example of this would be surrounding a similar access pattern to Figure 7.

Assume there are no links on the first page (P1) to the BBC homepage (P2). If the access pattern occurred frequently, then it can be seen that when the user visits bbccompetitor.com, they will subsequently visit bbc.co.uk and bbc.co.uk/sport. The fact that there is no link between these sites is significant in that Nanopoulos’ determination of frequent access patterns will only determine the pattern P2 to P3 and not {P1,P2,P3} or {P1,P2}.

Figure 7: Example access pattern to model Nanopoulos limitations
A solution does not currently exist that accurately models Chen’s maximal forward references, and a user’s cross-site access patterns. A new proposed adaptation to these approaches will need to be determined.

2.4.4 Implementation Languages

Existing approaches have used a large selection of implementation programming languages. From the limited information available on existing implementations, it is apparent that Letizia is implemented in Macintosh Common Lisp, WebMate is implemented in C and all the implementations use HTML as a means of interface interaction. By using HTML as a means of user interaction with the system, it is immediately obvious that a second language must be used to implement the system itself, so an adopted HTML interface requires an extra overhead of communication protocols to communicate between the HTML interface and the system.

If the system were implemented in the same language as the user interface, the communication overhead would be reduced. Another advantage of using a common language would be the re-use of code. A common code library of functionality could be shared amongst the two programs, minimising time scales for implementation and reducing margins for error trapping.

The two available platforms for implementing the system are Sun Microsystems’ Java and Microsoft’s .NET. A brief summary of each architecture is given below in Figure 8.

<table>
<thead>
<tr>
<th>.Net Architecture</th>
<th>Java Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designed to support multiple different programming languages. Currently, 30 languages support the .NET architecture.</td>
<td>Though other languages’ code can be converted to run under JVM, they don’t acquire true cross-language capabilities.</td>
</tr>
<tr>
<td>Compiles the source code to Intermediate Language (IL), which is itself a language.</td>
<td>Compiles the source code to Java bytecode, which by itself is not a language.</td>
</tr>
<tr>
<td>CLR implements a contiguous memory allocation algorithm.</td>
<td>JVM implements a noncontiguous memory allocation algorithm.</td>
</tr>
<tr>
<td>Compiles the source code twice during the process of converting to native code. Compiling works faster than interpreting.</td>
<td>Compiles and interprets the source code once during the process of converting it to native code.</td>
</tr>
</tbody>
</table>

Figure 8: Summary of .NET and Java architectures [INFORMIT]

A key finding of the .NET is to encourage cross-language development. The advantage is that the developer can choose the language that best suits for delivering a given module/unit (each language has its own strengths) and still be able to integrate into a single application. Code reusability, support for web services and platform independence are three strategic considerations for an implementation language. Each feature is identified below in Figure 9.
2.4.5 Usability

Establishing an understanding of HCI issues in the context of web users is an important area of discussion. Hermans [Her96] identifies the high-prioritised level of usability required for user interaction specifically with an agent. He suggests that users will not start to use agents because of their benevolence, proactivity or adaptivity, but because they like the way agents help and support them in doing all kinds of tasks. Therefore, a major issue is that the agent should suit the needs of the user, but in particular, a web user, in an extremely utilizable approach. Nielsen’s proposed “Ten Usability Heuristics” [10USAB] are a firm starting point:

1. **Visibility of system status**
   The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

2. **Match between system and the real world**
   The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

3. **User control and freedom**
   Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

4. **Consistency and standards**
   Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. **Error prevention**
   Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

6. **Recognition rather than recall**
   Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

7. **Flexibility and efficiency of use**
   Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8. **Aesthetic and minimalist design**
   Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. **Help users recognize, diagnose, and recover from errors**
   Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. **Help and documentation**
    Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.

With regard to web users and the possible interaction with an intelligent agent, some other issues require addressing. Lieberman [Lie97] identifies that recommendations from a web browsing agent are never critical decisions, so providing suggestions and recommendations can only benefit a user’s browsing activities, given that the agent’s recommendation are somewhat worthwhile. Providing recommendations rather than acting (i.e. changing to a recommended document without asking the user) offers a cooperative partnership between the user and the agent. This type of partnership would allow the user not to fear the recommendations that the agent makes will be forced upon them. Like Letizia [Lie97], a tool should not insist on acceptance or rejection of its advice.

When evaluating the effectiveness of the interface of the agent, these issues should be taken into consideration and addressed where possible.

### 2.5 Summary

In this section, a brief introduction to the notation of intelligent agents and the existing techniques used have resulted in a comparison of the system architectures, learning methods, implementation languages and HCI considerations necessary to determine a specification for a set of smart web browsing tools. Decisions on these aspects for a new approach to smart web browsing tools are to be addressed in the next section.
3 Decisions and Technical Challenges

This section identifies the important decisions in the scope of the project and the technical challenges involved in implementing these decisions. New techniques and approaches to the problem are proposed and contrasted with those discussed in the background of existing research.

Discussions in the previous section of this report outline that decisions involving the set of tools to employ, the web mining techniques to utilise, the system architecture to house these tools and the usability of the implementation need addressing.

3.1 Set of Tools

The discussion of the existing tools concluded that currently there does not exist a toolset, i.e. a collection of tools available to web users through a single user interface. A collection of such tools working in unison for both novice and experience web users is clearly sought after. Therefore I propose incorporating the following three tools into a single end user interface.

3.1.1 Personal Shortcut Recommendations

The discussion of existing tools identified that there currently exists a tool (Footprints [Wex99, Wex97]) that recommends shortcuts to users to a web site, these recommendation are due to the access patterns of other users. A tool which provides recommendations of shortcuts to popular pages in real time to a given users based on the access patterns of that user has not been achieved. The tool described assists users in browsing web sites they commonly visit in a more effective fashion.

3.1.2 New Document Recommendations

Similar tools have provided effective results in the past, each in a different manner. The first method from Letizia [Lie01, Lie97, Lie95] uses forward prediction from a user’s current location. However, this technique is proportionally effective with regard to the bandwidth available to forward search. The technique used by PowerScout [Lie01] and Almathaea [Mou97] uses search engines recommending results based on automated queries to them. This method has the advantage of working better on a low-bandwidth internet connection as well as high-bandwidth. In addition, prefetching system typically has only a limited amount of time to prefetch before the user makes a new selection. The response of querying a search engine is far quicker than this type of manual searching, offering shorter delays for a user to receive recommendations, a crucial aspect of a web assistant. Given that novice users are unfamiliar with a search engine query interface, a tool that queries search engine on their behalf abstracts and reduces the learning curve of understanding search engine.
3.1.3 Bookmark Recommendation

Novice users take a long time to realise the benefits of bookmarking popular sites and require nurturing from an early age their learning experience on the advantages of bookmarking. Even experienced web users may not be using bookmarks to the usable extent they should. Therefore both sets of users would benefit from a tool that recommended popular pages they visit as bookmarks.

3.2 Web Mining Techniques

The toolset decided from 3.1 requires consideration of autonomy, document representation, user interest modelling and access pattern modelling.

3.2.1 Autonomy

The discussion of existing approaches concluded that it is currently unproven that a ratings-based learning approach performs better than an autonomous machine learning approach to intelligent web agents. However the usability, performance and consistency aspects of the autonomous learning have proven to be dramatically superior in a non-ratings based approach. Therefore, I propose employing an autonomous approach to learning a user’s interest based on agent adaptation and not user ratings. During learning, the level of burden on the user would be kept to an absolute minimum.

3.2.2 Document Representation

Documents (i.e. web page) viewed by web users have been successfully modelled using the Vector Space Model [Sal83] by almost all the existing approaches. Davison [Dav02] ascertains that textual similarity–based predictions (using a document vector) outperform the simpler (mathematical) approaches. Using this model, a document is represented by a (term, weight) vector, where the weight of a given term is calculated using TF-IDF formula. Similarity between documents is calculated using a cosine similarity measure.

Representing a document in the Vector Space Model does lose the document hierarchy and therefore overall accuracy up to an extent. However, it allows a range of other techniques to be employed on the document vector. Documents would subsequently be stemmed [Por80] as performance difference in storing and accessing a small list of terms outweighs the marginal loss in document accuracy.

Instead of storing and accessing the complete document vector, the top N terms (to be determined) ordered by weight would best represent the vector. Above this number of terms, the weight of the term would be negligible compared to the other terms in the vector and would not pose any advantages during the cosine similarity comparison. Instead, the cropped vector would denote fewer computational comparisons when performing cosine similarity and would allow an efficient document representation.

Research [Kar00, Bal95] demonstrates that normalising the weight of each term in a document vector leads to a dramatic improvement in document representation quality. Raw similarity between each term $u$ and its $k$ most similar documents can be significantly different due to TF-IDF weight value generated will be a raw value for its occurrence, not a percentage value. This happens when a document is of significantly different length to
another during comparison. For example, if a particular term $t$ occurs in 10% of a document (length 100 terms) and this is compared with a document of same occurrence percentage by length 1000 terms, the TF-IDF weight generated would give a $t_f$ a value 10 times as small as the second document. By normalising the weights of each term with the number of cropped items in the vector, a truer reflection of the importance of each term would be stored.

### 3.2.3 User Interest Modelling

#### Categories of Interest

Web users do not have a single category of interest. Therefore, it is necessary to model a user interest profile using several categories of interest. A category is defined as a set of documents where the similarity between each document vector and the category vector is above a given threshold similarity. Figure 10 illustrates how a set of documents would be modelled in the Vector Space Model.

![Figure 10: How multiple documents would be represented in a category collection of similar documents in the Vector Space Model](image)

![Figure 11: How an irrelevant document would be represented with regard to a category in the Vector Space Model](image)

An outlier can be defined as a document that differs strongly from the documents in a given category. Therefore, when a document is viewed by a user, but does not relate to an interest category in both the short and long term, the document represents an outlier. This is a randomly accessed page of non-relevant information. When an irrelevant page is accessed and has its similarity tested against the current set of cluster categories, the result will be well below any similarity threshold value, as only a negligible number of terms, if any, will match and signal a similarity. Therefore, it can be assumed that an outlier will not to belong to any current categories of interest, as in Figure 11.

A user’s interest in a topic is rarely represented by a single theoretically optimal document. Therefore, the centroid of a category is more accurately represented by the average of the document vectors contained within it, than a mediod approach as long as the category has adequate protection from outliers. Therefore, the centroid ought to be represented by the average of the document vectors.

#### Document Incorporation

Advertisements and irrelevant information in the document can lower the accuracy of the system [Che98]. These outlier pages represent two possible user behaviour during document incorporation. Firstly, the document could be a single access that the user has stumbled across, which in itself can be seen as an irrelevant document with regard to the user’s interest and thus would never access a similar document again. Secondly, the page
could signal a change in the user’s interest profile, as a new interest does not necessarily have to overlap with a previous interest.

Previous user profiling algorithms have provided two distinct solutions to this problem. The first solution is to disregard this document as it does not match a particular category. The problem of a single access is addressed as the document is not incorporated into the profile for the right reasons. However, a potentially new interest category will be consistently disregarded as the document under contention will not overlap with a previous category. This means that a user interest profile is relatively inflexible to sudden change of browsing interests. However, this solution does provide for a more efficient algorithm, as a single document category will not exist in the profile and will not need to be computed.

The second solution, from [Che98], will always incorporate a new document into a category averaging the two most similar vectors if there is not enough space in the cluster. The significance of this is that irrelevant documents will be included in the category and thus will have some negative effect of the representation of the user profile. However, as new documents are included as standard, this solution proves to be more lenient with generation of new distinct categories.

No available solution exists to handle both outliers as well as new, multiple category interests. A new proposal for modelling multiple category user interests is therefore required and subsequently defined.

**Proposal Defined**

Before explaining how a page is incorporated, it is necessary to briefly redefine the elements in the interest profile space model.

- A document is defined as a top-N cropped term vector where the weight of each term is modelled using a normalised TF-IDF weight measured.

- A category of interest is defined as a set of documents where the similarity between each document and the category centroid is above a given threshold. The category centroid is represented by the average of all the documents belonging to the category.

- An irrelevant document is defined as a document vector having a similarity comparison below a given similarity threshold of a given category.

The new proposal will use a new principle known as a “Gutter” to handle problem. This principle allows irrelevant documents to be temporarily held in the Gutter to be given the chance to prove it’s long term worth. The theory behind the Gutter in the proposed algorithm is to provide a means of allowing documents a predefined amount of tolerance to distinguish whether the document represents a new category or an irrelevant page. This is achieved by adding these irrelevant documents to the finite length buffer called the Gutter. Each time a document is added to the Gutter, the newly orphaned document is compared with other orphaned documents in the Gutter, in the hope that if this document is similar to other orphaned documents, then these documents would accurately represent a new category of interest. These documents are then removed from the Gutter, promoted together to form a new category of interest. This allows short term orphaned documents a finite chance to find similarity with other documents representing the new category.

The Gutter has a finite length to identify long term orphaned documents. If the Gutter reaches its capacity, subsequent additions to the Gutter will cause the oldest orphaned
document to be disregarded. This should be considered satisfactory, as an orphaned document that does not match with other orphaned documents in the Gutter over a finite period of time, has a high probability of being a long term orphan and therefore should not be included in future similarity checks.

An explanation of the pseudo-code from Figure 13 follows for each group of line numbers:

### 1-7 Category Incorporation

The new document vector is primarily compared with each centroid vector each of the existing categories of interest. If the new document is similar to one or more categories, the new document is added to each category’s collection of document and a new centroid is determined for each category. The centroid is the average vector of all the document vectors existing in the category. The centroid is not cropped like the document vectors. Instead, it can grow to a maximum of:

\[ V_{\text{max}} \times D_{\text{max}} \]

Where \( V_{\text{max}} \) is the total number of terms in a document and \( D_{\text{max}} \) is the total number of documents in the category set. Each category that has incorporated the new document is added to a list of changed clusters.
8-21 **Gutter Demotion**
If the new document is not added to any categories (this is recognized by checking the size of the changed clusters list), then the new document is (in the short term) irrelevant. Therefore, the new document is added to the gutter.

Otherwise, the new document was added to at least one category cluster, so checks must now be performed on the number of documents in the category. If the category has more than the maximum number of documents, the least similar document to each category centroid is determined. If this document is unique, i.e. doesn't exist in any other categories, then this least similar document is demoted to the gutter and a new centroid determined for the category cluster.

22-24 **Determine New Categories**
This part of the algorithm determines any new clusters to be formed from the gutter, by applying a K-mean algorithm to the gutter.

25-32 **New Category Promotion**
Check each potential cluster generated by the K-mean algorithm and promote any to a new category which has at least the required number of documents to form a new cluster. Each time a new cluster is promoted, the unique documents from that cluster (i.e. not in any other potential new clusters) are removed from the gutter.

33-35 **Gutter Garbage Collection**
This part of the algorithm applies garbage collection to the documents in the gutter. The oldest documents are removed from the gutter until it hold at most the maximum number of documents.

36-38 **Category Set Garbage Collection**
This part of the algorithm applies garbage collection to the clusters. The oldest clusters are removed from the cluster set until it holds at most the maximum number of clusters.

```plaintext
1  Foreach (Ci in C) {
2     If (sim(DNEW, Ci) > CSIM && !ExistsInCi(DNew)) {
3         Add DNEW to Ci
4         Determine new centroid Ci'
5         Add Ci' to CH
6     }
7  }
8  If (CH is empty) {
9      ADD DNEW to gutter G
10  }
11 Else {
12     Foreach (Ci' in CH) {
13         If (|Ci'| > KMAX) {
14             Determine least sim(DL, Ci)
15             If (DL does not exist in C) {
16                 Demote DL to G
17             }
18             Determine new centroid Ci'
19         }
20     }
21 }
```
Foreach (Dg in G) {
    Apply K-mean algorithm (with overlap) to Dg forming new clusters in GC
}

Foreach (GutterCluster in GC) {
    If (|GC| > KMIN) {
        Foreach (Dg in GutterCluster) {
            Remove Dg from G
        }
        Insert GutterCluster into G
    }
}

While (|G| > GMAX) {
    Remove oldest Dg added to G
}

While (|C| > CMAX) {
    Remove oldest (last updated) Ci in C
}

Figure 13: User interest profiling algorithm pseudo-code

Additional notes on the incorporation algorithm include:

**Old interests will eventually be removed**
The oldest interest category to be updated has the least probability of being an interest in the future. The latest interest category updated has the highest probability of being an interest in the future. Therefore, when the number of categories becomes too large, the oldest (last updated) category is removed and the new category takes its place.

---

### 3.2.4 Access Pattern Modelling

For the purpose of access pattern determination, each document should be considered as atomic, i.e. its structure is not taken into account. The ordering of accesses inside the sequential patterns is important, drawing a clear distinction with general association rules where for large itemset generation, ordering is not taken into account.

**Determining the Starting Database Set**

Chen’s [Che98] original maximal forward (MF) path algorithm provided an accurate representation of the starting database required to determine access patterns in the context of the web. However, the way Chen dealt with repeated forward references does need addressing. Chen’s algorithm takes the set of MF paths determined from a single given path and uses the unique paths only. For example, the path \{ABCBDDBC\} produces the MF path set containing \{ABC, ABD\}, removing the duplicate paths. In the context of web access patterns, an access pattern should include these duplications as their inclusion provides a more accurate representation of composite repeat patterns inside one web browsing session.
Subpath Containment
Nanopoulos [Nan01] identifies that support counting of subpath definition should preserve the ordering of accesses and addresses a limitation of Chen's section containment to noise. For example, Chen's section containment does not identify \{ABCD\} as a subpattern of \{AXBCYZD\}. Given that web browsing habits exhibit a large degree of noise (i.e. interleaved random access amongst a common access pattern), the definition of subpath containment should be used when modelling user access patterns.

Link Acknowledgement
Graph traversals in sequential web pattern mining refer to the available links from one document to another. Web users portray sequential access patterns across different web sites regardless of the availability of links between them. Therefore, an accurate model of web access patterns ought to ignore available graph traversals recommended by Nanopoulos [Nan01]. This proposal models documents in a true, connected web.

Important Itemsets
All frequent itemset are important to modelling access patterns, not just the final maximum itemset or its frequent subsets. 1-itemsets denote frequently accessed pages regardless of subsequent access. This will be used for identifying popular single pages, i.e. bookmark recommendations. Subsequent itemsets denote shortcuts, for example if the access pattern \{ABCD\} is frequent, then it is clear that if a user visits document A, they are likely (with a predetermined minimum support) that they will visit B, C and most importantly D. D represents the maximum-length shortcut in this instance. Whilst this maximum frequent itemset represents the maximum shortcut available to a web user, shortcuts to B and C are also important to the user, and should be recommended to the user. The reason for this usefulness is described using an example where:

- Document A represents a home page to a news web site
- Document B represents a business news page
- Document C represents a sports news page
- Document D represents a home page to an email web site
- Documents W, X, Y and Z represent business and sports news articles (infrequent)

If a user has a frequent access pattern consisting of ABCD and the user undertakes a pattern consisting of ABWXCYZD, then simply advising them to shortcut to D when they view A should not be the only advice. The user would obviously be interested in a shortcut to the business news page B, then after viewing business news articles on varying pages (W,X) would subsequently be interested in a view the sports page C. Only after viewing sports news articles on varying pages (Y,Z) would the user be interested in checking their email on page D. Therefore, frequent subsets should also be tracked and recommended to web users.

Real-Time Access Pattern Modelling Proposal
In order to model access patterns in real-time so that user’s can be given accurate, personal shortcut advice, I propose an adaptation to Nanopoulos’ frequent sequential path itemset generation algorithm, illustrated in Figure 14.
An explanation of the pseudo-code from Figure 15 follows for each group of line numbers:

### 1 Stop Repeat Refresh Abuse
The first part of the algorithm checks to see that the new page is not the same as the last page viewed. This is called a repeat refresh. Repeat refresh abuse can lower the accuracy of an access pattern, even if the user is unaware of this happening.

### 2 New Session Indication
A browsing session is defined as a continuous set of page accesses. Two instances would instigate an end to a web browsing session. The first is denoted by a predefined time difference between the current page viewed and the last page viewed. This is typical a start to a new browsing session. The second is denoted the buffer reaching a predefined limit. This is due to storage and computation limitations that can be imposed on the access pattern.

### 3-10 Session Promotion
The log is defined as a finite list of previous browsing sessions. The current buffer of page accesses is converted to a session instance. This session is added to the log and the buffer is subsequently emptied. If the log is full (which is assumed to happen at some point), then the oldest session is removed. This provides for accurate flexibility that a user access pattern changes with time. Old browsing habits will be phased out, and new browsing habits make a new impact if they are continued. The log now contains a more recent representation of browsing sessions of the user, since a new session has been added. Therefore, a set of new frequent access pattern is generated using the log.
11 Buffer Addition
If a recalculation of frequent access patterns has taken place, the new page is added to the empty buffer as the start of a new session. If the page doesn’t represent the start of a new session, then it is appended to the end of the buffer (the buffer will contain some pages).

12 Last Page Set
The last page viewed is set to the new page in anticipation of the next page incorporation.

IncorporatePage(PageNEW) {
    1. if(PageLAST.ID != PageNEW.ID) {
    2.     if(PageNEW.TIME–PageCURRENT.TIME > TTHRESHOLD OR Buffer.Size==BMAX) {
    3.         Buffer = SessionNEW
    4.         Log.Add(SessionNEW)
    5.         Buffer.Empty()
    6.         if(Log.Size > LMAX) {
    7.             Log.RemoveOldestSession()
    8.         }
    9.         RecalculatePatterns(Log)
    10.     }
    11.     Buffer.Add(PageNEW)
    12.     PageLAST = PageNEW
    13. }
}

Figure 15: Pseudo-code for real-time sequential frequent itemset generation algorithm

The RecalculatePatterns method involves an adapted form of Chen’s maximal forward path algorithm and an adapted form of Nanopoulos’ subpath generation algorithm. An explanation of the pseudo-code from Figure 16 follows for each group of line numbers:

1 Generate Maximum Forward Paths
As discussed earlier in this section, the database is set up after determining the set of maximum forward paths (virtual sessions) from the actual sessions in the log.

2-3 Generate Initial Set of Paths and Initialise k
The initial $C_1$ set of paths is generated to start the algorithm, where:

$$C_1 = \text{The set of all paths of length 1, } \forall c \in C_1, c.count = 0$$

K is also initialised to 1 for the subsequent while loop.

4 While Definition
Repeat lines 5-13 until $C_k$ (generated from $L_{k-1}$) is empty, i.e. no more frequent itemset exist for this or subsequent values of k.

5-10 Subpath Generation [Nan01]
The set of Subpaths is generated such that:

$$\text{Subpaths} = \{\text{subpath } | \text{subpath } \in C_k, \text{subpath } \in \text{Session} \}$$
A subpath of a Session = \{session_1, ..., session_n\}, Subpaths = \{subpath_1, ..., subpath_n\} for which
- Subpaths is also a Session and
- There exists m integers i_1<...<i_m such that
  subpath_1=\text{session}_{i_1},..., subpath_m=\text{session}_{i_m}

For example \{ABD\} is a subpath of the session \{ABCD\}, as is \{ABC\},\{ACD\} and \{BCD\}. Therefore noise (in the form of random page accesses does not affect the support of an access pattern.

11 Prune
The set of subpaths is pruned according to:
\[ L_k = \{ \text{subpath} \mid \text{subpath} \in C_k, \text{subpath.\text{count}} \geq \text{min.\text{Support}} \} \]

\( L_k \) represents the list of subpaths whose support is above the minimum support and each subpath exists in the database.

12 Append Patterns
Items in the itemset \( L_k \) are appended to the frequent access patterns list. This allows all important itemset (not just the final itemset) to be remembered, as discussed earlier in this section.

13 \( k+1 \) Candidate Generation
The candidates for the next phase of the algorithm are generated, based on the natural join of \( L_{k-1} \) with itself. For example if \( L_2 = \{AB, BC, CA, DC\} \), the natural join of \( L_k \) with itself would give \( C_3\{ABC, BCA, CAB, DCA\} \)

RecalculatePatterns(Log) {
  1. DB = GenerateMFs(Log)
  2. \( C_k = \text{GenerateC1(DB)} \)
  3. \( k=1 \)
  4. While (\( C_k \neq \emptyset \)) {
      5. \( \text{foreach(Session in DB) { } } \)
      6. \( \text{Subpaths = genSubpaths(Session)} \)
      7. \( \text{foreach(subpath in Subpaths) { } } \)
      8. \( \text{subpath.\text{count}}++ \)
      9. \} \)
  10. \( L_k = \text{RemoveMinimumSupport(Subpaths)} \)
  11. \( \text{FrequentPatterns.Append}(L_k) \)
  12. \( C_{k+1} = \text{genCandidates}(L_k) \)
  13. \} \)
}

Figure 16: Pseudo-code for the generation of sequential frequent itemsets

An example of the algorithm is illustrated using Figure 17. Note that the minimum support for this example is set to 2. Also note how session 4 in the log is rewarded as two
maximum forward paths when generating the database, before the sequential patterns are determined.

Figure 17: Example of sequential frequent pattern generation

Figure 17 would generate bookmark recommendations for documents A, B, C and D as well as shortcut recommendations of A→B, B→A, B→C, C→A, D→A, D→C. Note that the set {B, A} and {B, C, A} are frequent, however because they represent the same shortcut, on a single instance of this shortcut need be kept. This also applies to {D, A} and {D, C, A}.
3.3 System Architecture

3.3.1 Real-time Usage

The objectives of the project relate to personal assistance. This assistance must be tailored to and therefore based upon the interests and access patterns of that particular user. To achieve this, the system must provide real-time recommendations based on real-time user data, not web logs.

3.3.2 Information Collection

The discussion into web browser usage determined that the data tracking tool can be hard-coded into the browser, or can be tracked using a proxy server. The difficulty of modifying complicated browser code and the periodic shifts in browser usage both justify a proxy server approach to information collection. Facilitating a proxy server to collect browsing information allows cross browser compatibility, an essential feature in today’s fast changing browser market.

3.3.3 Distributed Architecture

A distributed architecture would provide for a far more complicated approach to the system. This approach requires an implementation of messaging between distributed components and increased error handling due to the haphazard nature of the network environment. However, these technical challenges allow a plethora of advantages to such a distributed system. These include:

- **Maximise resource usage** - If the intelligence component is place on a distributed location, the user can delegate processing power and bandwidth consumption to this distributed location allowing expensive calculations and searching to be undertaken remotely, with only the results delivered back to the user when necessary. This allows a maximisation of processing power and bandwidth usage for the user.

- **Increased flexibility for interface usage** – The persistent information of a web user will be stored in a remote, distributed location. This allows the user flexibility in that the user will be able to use the interface and therefore access their personal information in any location on the network. Therefore, the program is not tied to a single given computer.

- **Increase operating system independence** – The intelligence component could contain the majority of the implemented code so that the client interface component is as “thin” as possible. In this case, the interface merely acts as a message interpreter from the system, providing results and forwarding user requests to the intelligence component. Offloading the implementation weight to a distributed component allows a collection of client interfaces to be programmed quickly for any given operating system. Whilst the intelligence component will reside on a fixed operating system, the scope for client interface independence is increased.
3.3.4 Implementation Language

In the context of the project, four important advantages of Microsoft’s .NET framework over the Java framework include faster program execution, multiple cross-language support, true cross-language code reusability and familiar graphical user interface. Each of these poses different reward for implementation of the system.

Faster Program Execution
The importance of providing real-time recommendations to the user has been discussed in 3.3.1 - Real-time Usage. Any increase in speed of program execution is advantageous in facilitating real-time recommendations.

Multiple Cross-language Support
There are currently 30 different programming languages supported under the .NET architecture. Different parts of the system can be implemented in different languages, but with transparent communication between parts. This means the best-suited language can be employed to implement the different parts. In addition, future expansion possibilities are increased (see 3.3.5 below).

True Cross-language Code Reusability
Given the decision to model a distributed architecture, code reusability is important as common code will be required to interpret the data in the different components. Therefore, the advantage of cross-language code reusability decreases the possibility of repeated code.

Familiar Graphical User Interface
The Windows operating system dominates the end-user market. The .NET framework provides graphical user interface components that model existing end-user programs more closely than the Java framework, reducing the need for end-users to adapt to a less familiar environment. This provides an increase in usability for the end-user.

Java is thought to have an influential advantage over the .NET framework with regard to operating system independence. Java’s initial, "write once run anywhere" strategy has been superseded by acknowledgement that the strategy doesn't work with the newest framework development kits. In scope of the four important advantages the .NET architecture provides and the recent change in Java's cross-platform strategy, potential system benefits of implementation under the .NET architecture outweigh the potential benefits of implementation under the Java architecture.
3.3.5 Future Expansion Modularity

If the system were implemented in a highly modular design, it could be easily adapted and expanded in the future to accommodate new tools and subsequent changes in web browsing habits. Designing and implementing a solid extendable framework provides for a more technically challenging approach to the project. However, the rewards for this approach, echoed in future ease of development, are important for the future usage and should be addressed. This approach coincides with an implementation under the .NET framework as possibilities are increased for future expansion to the system in another programming language.

3.4 Usability

3.4.1 Recommendation Strategy

Lieberman’s [Lie97] proposal to provide ignorable recommendations would offer a system with a higher degree of usability than an approach that explicitly acted on browsing suggestions and jeopardised a user’s trust in the system. Therefore a recommendation strategy which implies as un-obtrusive advice as possible should be implemented.

3.4.2 Usability Heuristics

Adopting Nielsen’s "Ten Usability Heuristics" [10USAB] offers a guideline for providing a highly usable interface as well as a concrete approach for evaluating the final product in relation to usability.

3.5 Summary

In this section the technical challenges required to fulfil the objectives of the project have been determined and the necessary decisions made to address these challenges settled in the theoretical sense. These decisions, backed by new proposals for modelling web browsing habits form the basis of the formal specification in the next section.
4 Specification

This section contains the specification for the implementation of the smart web browsing tools. The specification formally states the decisions raised in the previous section and specifies the core tools to be implemented.

The intended software result of this project is a fully functional end-user system which adheres to the following implementation requirements:

4.1 General Requirements

G1. The system will be implemented in three distinct programs: A proxy server program, an intelligence server program and an end-user client interface program.

G2. The implementation should provide an end-user with intelligent recommendations to personal web browsing habits and interests.

G3. The implementation should provide an end-user interface for recommendations to be displayed to the user and requests from the user to be processed.

4.2 Set of End-User Tools to Implement

T1. A tool should be provided to recommend shortcuts to pages a user regularly visits.

T2. A tool should be provided to recommend new web pages that match a user’s current interest.

T3. A tool should be provided to recommend the setting of bookmarks for pages a user regularly visits.

4.3 Web Mining Technique Adoption

M1. Acquisition of user data and learning should be achieved completely autonomously by the system.

M2. User interests will be modelled using the new proposed approach.

M3. User access patterns will be modelled using the new proposed approach.
4.4 System Architecture Requirements

A1. The three programs should have the ability to work under a distributed environment, residing on different machines linked on a network.

A2. User browsing data will be collected using a proxy server.

A3. The user data monitoring should be easily deactivated.

A4. The system will be browser independent and work alongside any major browser.

A5. The proxy server program should be accessible by multiple users simultaneously.

A6. The system will provide real-time recommendation based on streaming, real-time user data.

A7. The intelligence server program should be accessible by multiple users simultaneously.

A8. Security policy should be adopted to maintain user information privacy on the intelligence server.

A9. The weighting of code in the client program should be kept to a minimum and the intelligence server program should incorporate the maximum.

A10. The client program should be implemented to work on the Windows operating system.

4.5 Implementation Language Requirements

I1. The system should be implemented using the .NET architecture.

I2. A cross-language common code toolkit should be implemented and used to implement the three distributed programs.

I3. The system should be implemented in a manner where future expansions and changes to the system are effortless.

4.6 Usability Requirements

U1. The client program should provide recommendations, whilst the end user maintains absolute control over browsing.

U2. Recommendations to the user must be made in an apparent, non-obtrusive form, with the ability to ignore recommendations as seamless.

U3. The client program should provide the ability to be easily deactivated whilst the system is still monitoring user data.
U4. The client program interface should appear familiar and easy to become accustomed to.

U5. The client program should adhere to Nielsen’s Ten Usability Heuristics [10USAB]
5 Design

This section details the transition from specification to outline system design. The section includes a brief presentation of the overall design decided upon, covering each of the main areas of the system in more detail.

The intended software result of this project is a set of smart web browsing tools incorporated in a system that can be used by multiple concurrent users. The system will be called and referred to as Damaranke. Based on the specification from section 4, the three applications are conceptually designed below and will be followed during implementation.

5.1 System Overview

Figure 18 illustrates a conceptual design of the interaction between the main components; the intelligence server application, the proxy server application and the client application. Red components and arrows imply applications and communication to be implemented.

![Figure 18: Conceptual system component interaction overview](image-url)
The design overview from Figure 18 satisfies specification requirements G1, G2, G3 and A1. The dataflow through the system will follow the sequential numbering from Figure 18. An explanation of the dataflow is given below.

1 LAN settings forward HTTP request traffic to the proxy server
Only HTTP requests require forwarding to the proxy server application as only these are used for the purposes of web browsing.

2-3 HTTP Forwarding
The proxy server forwards the HTTP request to the WWW and receives a resulting response.

4 HTTP Response
The response from the WWW is forwarded back to the browser and becomes visible to the user.

5-6 HTML Content Logging
The content of the requested document is saved to the database directly by the proxy server and a message sent to the intelligence server that a new request has been logged. This initiates the intelligence server to incorporate the new document.

7-8 Read user profile
The document to be incorporated and the current profile of the user who viewed the document is read from the database by the intelligence server. The new document is incorporated into the user profile. The changes to the user profile are subsequently saved to the database to maintain persistence.

9-10 Advice Generation
Changes to the profile may reflect changes to the advice given to the user. Therefore, new advice must be determined which could incorporate searches from the WWW.

11-12 Advice
Changes to the advice for the user are sent via messaging to the client application. Recommendation links from the client program are followed by the user, presenting the recommended content in a new browser window.

5.2 Proxy Server
Proxy server APIs are readily available, therefore only extensions to a proxy server require implementation. These extensions will provide the functionality to save the HTML content of a response to the Damaranke database and send a message to the intelligence server to initiate a learn request. Therefore HTTP browser request forwarding need not be modified. Figure 19 gives a closer insight into the design of the proxy server application.
With all HTTP requests being channelled through the proxy server, user data will seamlessly be acquired by the system, satisfying specifications M1 and A2. The nature of a proxy server allows multiple concurrent users alongside any browser, satisfying specifications A4 and A5. This design finally satisfies A3 by allowing a user to simple disable channelling of HTTP traffic through the proxy server by disabling the feature at the browser option level.

5.3 Intelligence Server

The intelligence server is designed to be split into components illustrated in Figure 20:
Figure 20 illustrates how the communication manager abstracts the distributed architecture specified by A1. The learner acts as a collection of learner modules. This modular design satisfies specification I3 of effortless future expansion, as future learner modules can be easily added to the collection. The advisor uses the same approach. The proposed user interest and access pattern models will be implemented as modules in the learner, satisfying specifications M2 and M3. The three proposed tools will be implemented as modules in the advisor, satisfying specifications T1, T2 and T3.

Incoming messages received from other distributed components are filtered by the communication manager and forwarded on to the relevant learner module. A thread is created to initiate the learn procedure in order to satisfy specification A7, allowing the learner thread to compute in the background while the communication manager pools for another incoming message. Once new advice is determined for a user by the advisor, it will be sent to the client application via the communication manager in real-time, satisfying specification A6.

The database manager will abstract method calls to the database. This allows simple changes to the database manager’s connection properties without changing the method calls from the other components. Advantages of this include the possibility of easily changing the underlying database, again reiterating specification I3.

5.4 Client Application

The insight in Figure 21 illustrates the designed simplicity required to implement the client application. Advice from the intelligence server is echoed in advices pages, updated on request from the server under the influence of a thread. This design of simply echoing advice determined by the intelligent server satisfies the specification A9, where the code weighting falls heavily on the server to allow the possibility of multiple clients to be constructed quickly and effortlessly.

The design allows a user action (i.e. clicking a recommended link) to change the visible web page and therefore browsing path. This satisfies the specification U1 with the user...
maintaining absolute control over the browsing path while an explicit user action signals a change, not a program decision. The distributed nature of the client design means that if the client program is deactivated (i.e. closed), messages intended to be sent to the client from the server will fail and the current advice will not be displayed to the user. However, if the proxy server and the intelligence server continue to run, user data will still be monitored even without the client application running. This satisfies specification U2.

The client application will be the only application with a graphical user interface, customised for the end-user. The interface design will mirror that of Figure 22. Descriptions in the diagram entail how each feature contributes to adhering specification U5, Nielsen’s 10 Usability Heuristics:

Some of the usability heuristics are not included in the design of the immediate interface as they concern error messages, error prevention, help and documentation. The client program will be implemented using Microsoft’s set of common controls, making the interface appear familiar to the windows environment, satisfying specification U4.

Authentication in the form of a username and password will require access to a given user profile. All functionality of the program will be disabled unless valid authorisation is achieved by the user. This satisfies the security specification of A8.
5.5 Toolkit

The unaddressed specification I2 concerns the implementation of a cross-language common code toolkit. This toolkit will contain a set of programmed objects that can be used to implement the three components. This toolkit will include:

- Database connection objects
- Distributed messaging objects
- Common abstract data types used by more than one component.
6 Implementation

This section details how the important parts of the system were implemented and describes any problems that had to be overcome during their implementation.

All components of the system were implemented using Microsoft C# language under the .NET framework. Details of each of the components are given below.

6.1 Toolkit

The toolkit is a set of reusable abstract objects designed to facilitate concise implementation of the three system applications. Objects included are detailed below.

6.1.1 User Interest Abstract Data Type

![Unified Modelling Language diagram of a user interest profile abstract data type](image)

Figure 23: Unified Modelling Language diagram of a user interest profile abstract data type
The user interest abstract data type was included in the toolkit in order to provide a simplified interface to model a user interest profile, illustrated in Figure 23. Abstract data types were generated for each of the components under the model including Gutter, Category, Centroid and Document abstract data types. This approach utilises the façade pattern [Coo01], providing a simplified interface to the user interest profile system by simplifying the complexity of functions appearing to the data type user. Notice the method in the Profile class named IncorporateDocument which abstracts the user interest profiling algorithm as a single method call.

Figure 24: Example of a two-term document user interest profile in XML form

Another important feature of this data type is the method ToXML() which appears in every class. This method generates an XML string [XMLORG] representation of the user profile illustrated in Figure 24, with delegation of the string generation as a chain of command pattern [Coo01], where each component class handles its own determination of an XML element. The output of this method can then be stored as a persistent profile in the
Implementation

database (6.5). A declaration method of the Profile class takes as input an XML string and reengineers the model. This method is used when a profile is read back from the database.

6.1.2 User Access Pattern Abstract Data Type

The user access pattern abstract data type was also implemented in a similar manner to the user interest profile type. Figures of the class diagram and an example of the XML representation are included in Appendix A.

6.1.3 Damaranke Messaging Service

The Damaranke Messaging Service (DMS) is a set of classes included in the toolkit to simplify the sending and receiving of messages between the distributed components in the system. They include:

**DMSMessage**

This class models the contents of a message, providing methods to set property values such as message type, source and contents. DMSMessage contains methods to convert the structure to and from an XML string like Figure 25 for use with the Sender and Receiver classes.

```xml
<DamarankeMessage Source='10.0.0.7" Type='getBookmarks'>
  ..messagecontents...
</DamarankeMessage>
```

Figure 25: Example of a DMSMessage as an XML string

**Sender**

This class encapsulates the sending of a DMSMessage as an XML string over TCP.

**Receiver**

This class encapsulates the receiving of a DMSMessage as an XML string over TCP.

6.1.4 Database Manager

The Database Manager is a set of classes included in the toolkit to provide connection properties and method calls to the database. This allows a modification of the database without change to the underlying code of the rest of the system.

6.2 Proxy Server Application

The proxy server application was based on an existing proxy server from Mentalis [MENTALIS]. The Mentalis proxy server is a console application shown in Figure 26, and was modified by implementing new DamarankeHTTPListener and DamarankeHTTPClient classes in order to provide a link to saving to the system database and messaging the intelligence server for a learn request.
6.3 Intelligence Server Application

The intelligence server was implemented as a console application, illustrated in Figure 27.

Other than the starting text, only error messages are appended to the console screen. This was done to improve the performance of the system as printing data to the console was found to take up computation time.

Important aspects with the implementation of the intelligence server are described below.

6.3.1 Intelligence Modules

A Learner class was implemented with abstract methods to provide a modular approach to the intelligence server. Figure 28 illustrates how the Learner class includes an abstract method “learn”, defined by each sub-classed module.
Each abstract learner class is held in a collection of learner objects. When message is received from the proxy server initiating a learn request, every module’s learn method is called determining a different sequence of events for each module. With an addition of a single line of code in the server application, intelligence modules can be added to the system as long as they sub-class the provided Learner class. The same technique applies to the set of advice modules with an abstract Advisor class defined in the system.

6.3.2 User Interest and Access Pattern Caching

User profiles and access patterns are stored in the database as XML strings [XMLORG]. During document incorporation in both the profile and access pattern algorithms, persistent data is required to store the changes.

During implementation it became apparent that accessing the database and reengineering the user interest and access pattern profiles from XML with every document incorporation resulted in inadequate system performance. To overcome this problem, a caching technique was adopted. In this technique, a cache of the last N profiles are held in memory and calculations performed with these.

If a user interest or access pattern profile is not in the cache, it will be read from the database first. If the cache already contains N profiles then the least recent profile will be saved to the database and removed. The value of N is program definable and is chosen depending on the estimated number of concurrent users of the system. A cache with too few profiles in comparison to current users will cause frequent thrashing, wasting time copying in and out of the cache. If a sensible value of N is used, the number of requests to the database will be significantly minimised and would respond with a decrease the overall computation time. In addition to saving a profile when removed from the cache, another thread running in the system saves all the profiles in the cache to the database after a
Implementation

given period of time. This safety measure is provided so that should the server application crash, there still a recent version of the profiles stored in the database.

6.3.3 Content Extraction

Originally, a set of regular expressions were used to strip the HTML from a web document in order to extract the content. However, this approach proved slow (regular expressions should not be used for parsing large documents) and erroneous (a large volume of documents contained invalid HTML markup). To overcome this problem, the content of a document was instead parsed using an HTML parser from MIL [MIL].

6.3.4 Google Recommendations

Recommendations of new pages, based on a user’s learned interests were found using the Google API [GOOAPI]. This API provided by the Google allows up to 1000 direct queries to their search engine per day. Using the API removed the need to query a search engine using standard HTTP protocol, where results obtained from Google would otherwise need to be parsed from the HTML.

6.4 Client Application

The client application was implemented as a Windows application, illustrated in Figure 30.

![Figure 30: Implemented client end-user interface windows application](image-url)
Important aspects of the implementation of the client application are described below.

### 6.4.1 Recommendation Alerts

The way in which recommendations were made to users was an important part of the implementation of the client application. This is due to research findings resulting in specification U2, which stated “Recommendations to the user must be made in an apparent, non-obtrusive form, with the ability to ignore recommendations as seamless”.

![Image of Recommendation popup message windows](image)

The resulting recommendation strategy took inspiration from a program familiar to a large share of windows users. MSN Messenger [MSNMESS] uses popup windows in the corner of the screen to alert users of the status of the program. A similar technique was implemented for the client application, with the popup message windows illustrated in Figure 31. Fading techniques ensure that these popup messages appear apparent, yet non-obtrusive, with the messages disappearing after a short period of time allowing them to be easily ignored.

### 6.4.2 Addition of My Bookmarks Tool

During implementation, it became apparent that recommendations for bookmarking a particular site would be repeated even if that page was actually bookmarked in the browser. This is due to a lack of communication between the client application and the browser. Two ways to resolve this problem existed. The first solution was to provide browser specific code to add the bookmark to the browser. However, this clashed with specification A4, which stated that the system should be browser independent. Therefore this was not an option. Instead, the solution that was actually implemented was to add a bookmarking tool. Bookmark tracking became an additional personalised feature. This meant that if a user added a page as a bookmark, this information was added to the database and became persistent. From this point onwards only bookmark recommendations that are not part of the tracked “my bookmarks” are recommended to the user.

### 6.4.3 Security

Security against user privacy was achieved by a username/password login mechanism to initiate the software. Access to all features of the software is disabled until authorisation is achieved. An invalid combination of username/password is met by a message box shown in Figure 32.
6.5 Database

A database was implemented in Microsoft SQL Server 2000 [MSSQL] to store data of:
- A user’s profile of interests, access patterns and bookmarks.
- User authentication information.
- Web page logs.
- Term frequency and document frequency estimation.

6.5.1 Schema

The schema of the database implemented is illustrated below in Figure 33.
6.5.2 Stored Procedures

All access to the database from the Database Manager (6.1.4) was implemented using stored procedures [MSSQLSTO]. A list of the stored procedures implemented is included in the Appendix D. An example of an implemented stored procedure is illustrated in Figure 34.

```sql
CREATE PROCEDURE saveAccessPattern
    (@accesspattern_id int,
     @xml text)
AS

BEGIN TRANSACTION

UPDATE Profile_Users
SET ACCESSPATTERN_XML=@xml
WHERE PROFILE_ID=@accesspattern_id

COMMIT TRANSACTION

GO
```

Figure 34: Example of a stored procedure in the database

The advantages of using stored procedures include:

**Single Call**
Stored procedures are a precompiled collection of SQL statements and optional control-of-flow statements stored under a name and processed as a unit. This means changes to the database only need to be mirrored by changes in the stored procedure and not the application code, allowing cross-language reuse of SQL statements.

**Faster Execution**
The stored procedure is compiled on the server when it is created, so it executes faster than individual SQL statements.
7 Testing

This section describes the test procedures for the implementation and the obtained results of them. These results then form the basis of a critical evaluation of the effectiveness of the web mining techniques and the usability of the system with regard to the specification laid out.

After implementation, a series of technical and subjective tests were undertaken to evaluate the underlying techniques behind the smart web browsing tools and the usability of the end-user interface.

7.1 Technical Testing

Technical testing was carried out on the system to ensure that the web mining techniques that the tools were based on were correctly implemented with the output being correctly determined.

7.1.1 Access Pattern Model

Recommendations to the user of bookmarks and shortcuts are based on the sequential access patterns web mining technique. A series of tests was carried out designed to expose potential flaws in specific targeted aspects of the web mining algorithm, then finally the overall algorithm. During these tests, a predefined set of simulated page accesses was the input to the tests. A summary of the results are given below, with the actual trace of the tests included section C.1 of the Appendix.

Maximum Forward Path Generation

The motivation behind this test was to confirm the correctness of the implementation of the maximum forward path generation algorithm.

<table>
<thead>
<tr>
<th>Input Log:</th>
<th>=== Log ===</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABCDCECBF</td>
</tr>
<tr>
<td></td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>=============</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Database:</th>
<th>=== Database ===</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>ABCE</td>
</tr>
<tr>
<td></td>
<td>ABF</td>
</tr>
<tr>
<td></td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>=============</td>
</tr>
</tbody>
</table>

The test results confirm an accurate implementation of the generation of maximum forward paths. Notice that the path ABCD appears in two sessions and is counted twice in the database. Also, notice the multiple length backwards references are accurate too.
Appropriate Subpath Generation
The motivation behind this test was to determine whether the generation of subpaths was
determined correctly. This test also attempted to expose the flaw of attempting to
determine k subpaths of a k-1 length log.

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>=== Log ===</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
</tr>
<tr>
<td></td>
<td>AB</td>
</tr>
<tr>
<td></td>
<td>=========</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output:</th>
<th>=== Subpaths Generated ===</th>
</tr>
</thead>
<tbody>
<tr>
<td>A:</td>
<td>1</td>
</tr>
<tr>
<td>B:</td>
<td>1</td>
</tr>
<tr>
<td>C:</td>
<td>1</td>
</tr>
<tr>
<td>=========</td>
<td></td>
</tr>
<tr>
<td>=== Subpaths Generated ===</td>
<td></td>
</tr>
<tr>
<td>A:</td>
<td>1</td>
</tr>
<tr>
<td>B:</td>
<td>1</td>
</tr>
<tr>
<td>=========</td>
<td></td>
</tr>
<tr>
<td>=== Subpaths Generated ===</td>
<td></td>
</tr>
<tr>
<td>AB:</td>
<td>1</td>
</tr>
<tr>
<td>AC:</td>
<td>1</td>
</tr>
<tr>
<td>BC:</td>
<td>1</td>
</tr>
<tr>
<td>=========</td>
<td></td>
</tr>
<tr>
<td>=== Subpaths Generated ===</td>
<td></td>
</tr>
<tr>
<td>AB:</td>
<td>1</td>
</tr>
<tr>
<td>=========</td>
<td></td>
</tr>
<tr>
<td>=== Subpaths Generated ===</td>
<td></td>
</tr>
<tr>
<td>ABC:</td>
<td>1</td>
</tr>
</tbody>
</table>

The test results signified that the subpath generation produces the correct subpath
generation. Notice when k=3, no subpaths are determined for the path AB.

Susceptibility to Noise
The motivation behind this test was to determine whether the resulting recommendations
would be susceptible to random, unique page accesses, something common to web users.
This test was therefore design to expose the algorithm’s susceptibility to noisy data. The
test comprised of two runs. The first run tested the patterns without noisy data, and then a
subsequent run included interleaved, unique noisy data. The output of the two runs was
then compared.

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No Noise</td>
<td>With Noise</td>
</tr>
<tr>
<td>Input:</td>
<td>=== Log ===</td>
</tr>
<tr>
<td></td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>ABCD</td>
</tr>
<tr>
<td></td>
<td>=========</td>
</tr>
</tbody>
</table>
The two test runs resulted in output data that was identical. The result of the tests was that the pattern determination was not susceptible to noise. Therefore, this signifies that the recommendations of the system are not affected by random, unique page accesses.

### Composite Pattern Detection

The motivation behind this test was to see if composite web browsing habits were detectable. Notice the input included two sets of interleaved composite patterns in varying order.

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>=== Log ===</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
</tr>
<tr>
<td></td>
<td>XYZ</td>
</tr>
<tr>
<td></td>
<td>AXBYCZ</td>
</tr>
<tr>
<td></td>
<td>XAYBZC</td>
</tr>
<tr>
<td>Output:</td>
<td>=== Frequent Patterns ===</td>
</tr>
<tr>
<td></td>
<td>A: 3</td>
</tr>
<tr>
<td></td>
<td>AB: 3</td>
</tr>
<tr>
<td></td>
<td>ABC: 3</td>
</tr>
<tr>
<td></td>
<td>AC: 3</td>
</tr>
<tr>
<td></td>
<td>B: 3</td>
</tr>
<tr>
<td></td>
<td>BC: 3</td>
</tr>
<tr>
<td></td>
<td>C: 3</td>
</tr>
<tr>
<td></td>
<td>CD: 3</td>
</tr>
<tr>
<td></td>
<td>BD: 3</td>
</tr>
<tr>
<td></td>
<td>D: 3</td>
</tr>
</tbody>
</table>

The output from the results confirmed that composite patterns are detectable by the algorithm.
Overall Test
The motivation of this test is to confirm that the combined tests produce an overall, accurate result.

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Log:</td>
<td>=== Log ===</td>
</tr>
<tr>
<td></td>
<td>ABCBC</td>
</tr>
<tr>
<td></td>
<td>DEF</td>
</tr>
<tr>
<td></td>
<td>ADUBECVWF</td>
</tr>
<tr>
<td></td>
<td>DAEXBFCYZ</td>
</tr>
<tr>
<td>Input Database:</td>
<td>=== Database ===</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
</tr>
<tr>
<td></td>
<td>DEF</td>
</tr>
<tr>
<td></td>
<td>ADUBECVWF</td>
</tr>
<tr>
<td></td>
<td>DAEXBFCYZ</td>
</tr>
<tr>
<td>Output:</td>
<td>=== Frequent Patterns ===</td>
</tr>
<tr>
<td></td>
<td>A: 4</td>
</tr>
<tr>
<td></td>
<td>AB: 4</td>
</tr>
<tr>
<td></td>
<td>ABC: 4</td>
</tr>
<tr>
<td></td>
<td>AC: 4</td>
</tr>
<tr>
<td></td>
<td>B: 4</td>
</tr>
<tr>
<td></td>
<td>BC: 4</td>
</tr>
<tr>
<td></td>
<td>C: 4</td>
</tr>
<tr>
<td></td>
<td>D: 3</td>
</tr>
<tr>
<td></td>
<td>DE: 3</td>
</tr>
<tr>
<td></td>
<td>DEF: 3</td>
</tr>
<tr>
<td></td>
<td>DF: 3</td>
</tr>
<tr>
<td></td>
<td>E: 3</td>
</tr>
<tr>
<td></td>
<td>EF: 3</td>
</tr>
<tr>
<td></td>
<td>F: 3</td>
</tr>
</tbody>
</table>

The result confirms the overall output is correct. Notice:

- The composite patterns ABC and DEF are correctly detected.
- The interleaved noise of items U, V, W, X, Y and Z is dealt with (ignored due to minimum support level).
- The input log contains the path ABCBC, and the output generates the extra support of the pattern ABC.

This concludes that proposed algorithm models technically models the user as defined.
7.1.2 User Interest Model

The implementation of the new proposal to modelling user interest was tested by a number of comparisons to original algorithmic approaches. Detailed traces are included in C.2 of the Appendix.

Comparison with Single Category of Interest

The motivation for this test is to compare the multiple category of interest aspect of the algorithm to the existing approach of single category of interest, facilitated in Lira [Bal95].

For this test, six pages were incorporated into a profile. The six pages were chosen to represent two categories of interest, i.e. two groups of 3 pages, where the 3 documents in a single group have high similarity between them. To simplify the test, only the top 3 terms were used to represent the document vector, and the weights were rounded to 3 decimal places.

One group of documents were web pages on weather and the second group of web pages were on the Xbox console. The comparison in centroids of the profiles is shown below:

<table>
<thead>
<tr>
<th>Existing Approach Output</th>
<th>New Approach Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Profile ID=&quot;40&quot;&gt;</code></td>
<td><code>&lt;Profile ID=&quot;40&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Category ID=&quot;1&quot; Name=&quot;&quot;</code></td>
<td><code>&lt;Category ID=&quot;1&quot; Name=&quot;&quot;</code></td>
</tr>
<tr>
<td><code>&lt;Centroid&gt;</code></td>
<td><code>&lt;Centroid&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Document ID=&quot;-1&quot;&gt;</code></td>
<td><code>&lt;Document ID=&quot;-1&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;bst&quot; Weight=&quot;0.085&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;bst&quot; Weight=&quot;0.161&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;classic&quot; Weight=&quot;0.043&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;east&quot; Weight=&quot;0.099&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;east&quot; Weight=&quot;0.099&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;fabl&quot; Weight=&quot;0.028&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;forecast&quot; Weight=&quot;0.043&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;forecast&quot; Weight=&quot;0.086&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;south&quot; Weight=&quot;0.103&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;live&quot; Weight=&quot;0.034&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;ninja&quot; Weight=&quot;0.044&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;ninja&quot; Weight=&quot;0.028&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;tailor&quot; Weight=&quot;0.051&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;tailor&quot; Weight=&quot;0.052&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;temperatur&quot; Weight=&quot;0.097&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;temperatur&quot; Weight=&quot;0.048&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;weather&quot; Weight=&quot;0.178&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;weather&quot; Weight=&quot;0.052&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;xbox&quot; Weight=&quot;0.271&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;xbox&quot; Weight=&quot;0.048&quot; /&gt;</code></td>
</tr>
<tr>
<td>&lt;/Document&gt;</td>
<td>&lt;/Document&gt;</td>
</tr>
<tr>
<td>&lt;/Centroid&gt;</td>
<td>&lt;/Centroid&gt;</td>
</tr>
<tr>
<td>&lt;/Category&gt;</td>
<td>&lt;/Category&gt;</td>
</tr>
<tr>
<td>&lt;/Profile&gt;</td>
<td>&lt;/Profile&gt;</td>
</tr>
</tbody>
</table>

The results illustrate how the new approach models the multiple categories in comparison to the existing approach. In the existing approach, “classic forecast” would be identified as a user interest, however in the new approach, the distinction between the two categories are maintained.
**Effectiveness of the Gutter**

The motivation for this test was to determine whether the Gutter theory provides a more accurate profile a user. The test involved an incorporation of a group of 3 similar documents, and two irrelevant documents. In the existing approach, irrelevant documents are instructed to form a new category automatically. The results below have been abbreviated for the purpose of this table, however the full detailed trace is included in the Appendix section C.2.2.

<table>
<thead>
<tr>
<th>Existing Approach Output</th>
<th>New Approach Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;Profile ID=&quot;41&quot;&gt;</code></td>
<td><code>&lt;Profile ID=&quot;41&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Category ID=&quot;1&quot; Name=&quot;&quot;</code></td>
<td><code>&lt;Category ID=&quot;1&quot; Name=&quot;&quot;</code></td>
</tr>
<tr>
<td><code>&lt;Centroid&gt;</code></td>
<td><code>&lt;Centroid&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Document ID=&quot;-1&quot;&gt;</code></td>
<td><code>&lt;Document ID=&quot;-1&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;bst&quot; Weight=&quot;0.161&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;bst&quot; Weight=&quot;0.161&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;east&quot; Weight=&quot;0.197&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;east&quot; Weight=&quot;0.197&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;forecast&quot; Weight=&quot;0.086&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;forecast&quot; Weight=&quot;0.086&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;south&quot; Weight=&quot;0.103&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;south&quot; Weight=&quot;0.103&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;temperatur&quot; Weight=&quot;0.097&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;temperatur&quot; Weight=&quot;0.097&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;weather&quot; Weight=&quot;0.356&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;weather&quot; Weight=&quot;0.356&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Gutter&gt;</code></td>
<td><code>&lt;Gutter&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Category ID=&quot;2&quot; Name=&quot;&quot;</code></td>
<td><code>&lt;Category ID=&quot;2&quot; Name=&quot;&quot;</code></td>
</tr>
<tr>
<td><code>&lt;Centroid&gt;</code></td>
<td><code>&lt;Centroid&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Document ID=&quot;781&quot;&gt;</code></td>
<td><code>&lt;Document ID=&quot;781&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;fabl&quot; Weight=&quot;0.167&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;fabl&quot; Weight=&quot;0.167&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;ninja&quot; Weight=&quot;0.260&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;ninja&quot; Weight=&quot;0.260&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;xbox&quot; Weight=&quot;0.574&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;xbox&quot; Weight=&quot;0.574&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Gutter&gt;</code></td>
<td><code>&lt;Gutter&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Category ID=&quot;3&quot; Name=&quot;&quot;</code></td>
<td><code>&lt;Category ID=&quot;3&quot; Name=&quot;&quot;</code></td>
</tr>
<tr>
<td><code>&lt;Centroid&gt;</code></td>
<td><code>&lt;Centroid&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Document ID=&quot;111&quot;&gt;</code></td>
<td><code>&lt;Document ID=&quot;111&quot;&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;financ&quot; Weight=&quot;0.195&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;financ&quot; Weight=&quot;0.195&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;movi&quot; Weight=&quot;0.201&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;movi&quot; Weight=&quot;0.201&quot; /&gt;</code></td>
</tr>
<tr>
<td><code>&lt;Term Term=&quot;yahoo&quot; Weight=&quot;0.603&quot; /&gt;</code></td>
<td><code>&lt;Term Term=&quot;yahoo&quot; Weight=&quot;0.603&quot; /&gt;</code></td>
</tr>
</tbody>
</table>

The result from this test illustrates how two categories of interest appear in the existing approach. These represent irrelevant documents. In the new approach, the irrelevant documents are held temporarily in the Gutter, to be promoted to a category status should only an additional relevant document be viewed in the short term.

The incorporation of an additional document that is similar to one of the documents in the Gutter illustrates how the new category of interest is formed.
It can be seen that in the new approach, the incorporation of the document into the Gutter has accurately determined that a new category should be formed. Though this forms the same category as the existing approach for category 2, the flexibility of determination of categories from the Gutter demonstrates that all necessary categories will always be formed.

If it is assumed that the irrelevant document in category 3 will remain irrelevant forever, then the affect of that irrelevant category in the profile of the existing approach does have an adverse effect. While the existing approach will permanently retain this irrelevant interest category (3), the new approach will include the irrelevant document in the Gutter. As recommendations are based on the interest categories, the irrelevant document in the Gutter will not be used to generate recommendations. In the existing approach, the irrelevant category would be included in the recommendations.

*Therefore, the new approach has the effect of increasing the precision of recommendations to the user, as only the user’s relevant interests are suggested, not the irrelevant ones.*

### 7.2 Usability Testing – Client Application

The client application acts as the end-user interface to the system, with the recommendations from the intelligence server echoed through this interface. Therefore how well the system acts as a set of smart web browsing tools is affected somewhat by the usability of client application. To evaluate the usability of the application, the accordance to Nielsen’s 10 Usability Heuristics is assessed in addition to comments and suggestions from a set of end-user testers.
7.2.1 Nielsen’s 10 Usability Heuristics

It is reminded that specification U5 stipulates that “The client program should adhere to Nielsen’s Ten Usability Heuristics [10USAB]”. Therefore, adherence to Nielsen’s 10 Usability Heuristics must be taken into account during the evaluation of the usability of the client application.

Each adherence to the 10 heuristics is discussed below, referring to the implemented interface illustrated in Figure 35.

1 Visibility of system status
The status bar displays the current status of the system as one of the three possible states: “Disconnected”, “Connecting” or “Connected”.

2 Match between system and the real world
The tabbing system acts to mimic the concept of pages to separate the different types of recommendations.
3 User control and freedom
Terminal functions are met with confirmation type message boxes.

4 Consistency and standards
Words and icons used are kept consistent so facilitate immediate common meanings.

5 Error Prevention
Disabling areas of the program when not applicable for use reduces the scope for errors to occur.

6 Recognition rather than recall
Icons are representative of the adjacent description, promoting recognition of icons rather than recall of description.

7 Flexibility and efficiency of use
Flexibility provided in that users can double click the element in the list or explicitly select and click the button in order to perform the same task.

8 Aesthetic and minimalist design
Simple and comprehensible list-type structure to model the collection of current advice.

9 Help users recognise, diagnose and recover from errors
Helpful errors explain the cause of the error and advice to prevent the error from occurring again.

10 Help and documentation
Help in the form of a user guide is easily accessible via the menu.

Examples of accordance to all of Nielsen’s usability heuristics imply an initial, firm assertion that an interface with sound usability has been successfully implemented.

7.2.2 End-User Usability Testing

It is known that determining how well a personalisation system works is a difficult task. This is due to the fact that subjective assessment would make up a large majority of the evaluation.

A second difficulty with regard to a smart web browsing system is that the usefulness of the system will change somewhat over time, as the system develops a more accurate profile of the interests and access patterns of the user.

Agent tools and their users are best described as a partnership. With all partnerships, time taken to develop it can vary. This means that unless the system was used over a long
period of time (many months), the results could be used as concrete evidence of its usefulness. Due to time considerations of the project, such extensive formative usability testing was not possible.

However, the system was used for the period of around a week by a diverse group of users every time they browsed the Internet. The breakdown of these users was as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Internet Experience Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shaminder Sansoy</td>
<td>21</td>
<td>High</td>
</tr>
<tr>
<td>Karan Lohia</td>
<td>20</td>
<td>High</td>
</tr>
<tr>
<td>Arash Mahdavi</td>
<td>19</td>
<td>Medium</td>
</tr>
<tr>
<td>Dimis Theodorou</td>
<td>22</td>
<td>Medium</td>
</tr>
<tr>
<td>Yannis Sakatis</td>
<td>55</td>
<td>High</td>
</tr>
<tr>
<td>Soulla Sakatis</td>
<td>49</td>
<td>Low</td>
</tr>
<tr>
<td>Andreas Michaelides</td>
<td>33</td>
<td>Low</td>
</tr>
<tr>
<td>Leonie Sakatis</td>
<td>16</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Figure 36: Breakdown of users who tested the software

Whilst a formal evaluation was not undertaken, discussions with these people did unveil some interesting findings.

All highly experienced internet users found the shortcut tool was most useful, both low experience users found the Google recommendations and most useful and the medium users were somewhat spread between the three. This leads to the belief that highly experienced web users portray a higher degree of repeated access patterns and find recommendations of a new page less useful to them.

Of the eight people who tested the software, six agreed that they would prefer to browse the internet with the system that not in the future. The two who disagreed, felt that the even though the system was useful and made the browsing time more effective, the fact that they had to initiate the client application in addition to the browser when they wanted to get advice was somewhat troublesome. The experience level of the two users in question was medium.

**7.3 Summary**

Technical testing has illustrated that there is positive reason to believe that the user interest and the access pattern modelling proposed in this project provide a stronger and more accurate representation of a user’s browsing habits. With a clear adoption of stringent usability guidelines, and a positive response from the limited end-user testing, portrays the strong indication that the usefulness and effectiveness of the tools are well above a satisfactory level.
This section concludes the report by looking back at the objectives set out at the beginning of the report and presents a set of thoughts and ideas of opportunities for future work.

The aims of this project set out at the start of the report were:

1. Research ideas and concepts of a system that could analyse user observed browsing activities automatically to underpin a set of tools that can help users to browse the websites they commonly visit in a more effective fashion.

2. Decide on the design of a system of smart web browsing tools and the underlying series of techniques that will be used to facilitate them.

3. Implement the system that meets the design criteria with sufficient reliability and quality that it can be released to, and used by, untrained web users.

4. Test and evaluate the system by the effectiveness of the web mining techniques and the usability of the implemented system of tools.

During this report, existing approaches to the concept of a smart web browsing tool have been compared and contrasted, with new proposals to a system architecture, user interest profiling and access pattern determination tested and evaluated as a success. Whilst the time considerations discussed, prevented a formal, quantitative evaluation of the effectiveness of the system, the technical underpinnings of the web mining techniques suggest that only subjective reasons would cause a user to find the recommendations completely ineffective.

Though the possibility was available to produce a set of isolated smart web browsing tools, great time and consideration was taken to ensure the design and implementation of highly reusable platform for the addition and extension of future tools. The cross-language toolkit implemented has been documented thoroughly in order to provide the maximum potential for the system’s future use. The result is a project that has attempted to go well beyond the experimental stage and produce a platform system that closely resembles a marketable end product.

8.1 Future Work

Whilst the implemented distributed architecture was technically more challenging than that of an isolated application, several areas for future work have opened up because of decision to adopt this type of design.

8.1.1 Collaborative Tools

The client-server architecture of Damaranke allows for an amalgamation of real-time, user web browsing data. Providing any form of tools that combine or make deductions between
different user data primarily requires that some form of data exists. Strong user interest and access pattern profiling has been achieved by this project. With this foundation in place, the future for a set of collaborative tools proves to be an attractive area for investigation. Cross-user deductions between similar user interest profiles or similar user access patterns could provide an additional set of tools with recommendations made in real-time but with an element of global popularity.

8.1.2 Experience Dependant Tools

During end-user testing, a finding sparked an idea into a new category of tools. The finding was that different types of users found different types of advice effective. This could allow a set of experience-dependant tools to evolve, where recommendations are based on an estimation of the experience level of the user. The estimation of the experience level of the user could be determined by a closer look at the broadness of interests, the time spent browsing the web, and possibly even an understanding of the type of advice the specific user prefers. This adaptive form of advice could further bridge the gap between the agent assistant, and the web user.

8.1.3 WordNet Incorporation

Improvements to the perception of the information worth of a page could be undertaken using WordNet [WORDNET]. This lexical reference system could be used to refine the content of a web page to draw more inference on the data than just keyword extraction. Words of dubious nature, such as "fire", have many definitions in the WordNet including "the event of something burning", "intense adverse criticism", etc. If an estimation of the definition of the word in the scope of the content could be determined, then similarity between documents containing a word of similar meaning could be adopted. This would not just use exact word comparison and could possibly provide a more accurate representation of the information contained in the content.
9 Bibliography

9.1 Publications

[Agr94] Rakesh Agrawal, Ramakrishnan Srikant
Fast Algorithms for Mining Association Rules
1994

[Arm97] Robert Armstrong, Dayne Freitag, Throsten Joachims, Tom Mitchell
WebWatcher: A Learning Apprentice for the World Wide Web
1997

[Bal95] Marko Balabanovic, Yoav Shoham
Learning Information Retrieval Agents: Experiments with Automated Web Browsing
1995

[Bou01] Christos Bouras, Agisilaos Konidaris, Eleni Konidari, Afrodite Sevasti
Introducing Navigation Graphs as a Technique for Improving WWW User Browsing
2001

[Cha99] Philip K. Chan
Constructing Web User Profiles: A Non-invasive Learning Approach
1999

[Che98] Liren Chen, Katia Sycara
WebMate: A Personal Agent for Browsing and Searching
1998

[Che98b] Ming-Syan Chen, Jong Soo Park, Philip S. Yu
Efficient Data Mining for Path Traversals
1998

[Che97] David Cheung, Ben Kao, Joseph Lee
Discovering User Access Patterns on the World-Wide Web
1997

[Coo01] James W. Cooper
Java Design Patterns – A Tutorial, Addison-Wesley
2001

[Coo97] R. Cooley, B. Mobasher, J. Srivastava
Web Mining: Information and Pattern Discovery on the World Wide Web
1997
[Dav02] Brian D. Davison
Predicting Web Actions from HTML Content
2002

Information retrieval: data structures & algorithms, Prentice Hall
1992

[Han00] Jiawei Han and Micheline Kamber
Data mining: concepts and techniques
2000

[Her96] Björn Hermans
Intelligent Software Agents on the Internet: an inventory of currently
offered functionality in the information society & a prediction of (near-
future) developments
1996

[Kar00] George Karypis
Evaluation of Item-Based Top-N Recommendation Algorithms
2000

[Lie01] Henry Liberman, Christopher Fry, Louis Weitzman
Exploring the Web with Reconnaissance Agents
2001

[Lie97] Henry Liberman
Autonomous Interface Agents
1997

[Lie95] Henry Liberman
Letizia: An Agent That Assists Web Browsing
1995

[Mia96] Dunja Mladenic
Personal WebWatcher: Design and Implementation
1996

[Mou97] Alexandros G. Moukas
Amalthaea: Information Filtering and Discovery Using A Multiagent Evolving
System
1997

[Nan01] A. Nanopoulos, Y. Manolopoulos
Mining Patterns from Graph Traversals Data and Knowledge Eng. (DKE),
vol. 37, no. 3, pp. 243-266
2001

[Por80] M. Porter
An algorithm for suffix stripping
http://www.tartarus.org/~martin/PorterStemmer/def.txt
1980
Bibliography

[Roc71] Jr. J. Rocchio
The Smart System – Experiments in Automated Document Processing
1971

[Rus95] S. Russell, P. Norvig
Artificial Intelligence: A Modern Approach, Prentice Hall
1995

Introduction to Modern Information Retrieval
1983

[Sch02] Ingo Schwab, Alfred Kobsa
Adaptivity through Unobstrusive Learning
2002

[Som99] Gabriel Somlo, Adele E. Howe
Agent-Assisted Internet Browsing
1999

[Wex99] Alan Wexelblat, Pattie Maes
Footprints: History-Rich Tools for Information Foraging
1999

[Wex97] Alan Wexelblat, Pattie Maes
Footprints: History-Rich Web Browsing
1997

9.2 Web Sites

[10USAB] Jackob Nielsen’s Ten Usability Heuristics
http://www.useit.com/papers/heuristic/heuristic_list.html

[DOTNET] Microsoft .Net Information
http://www.microsoft.com/net/

[GOOAPI] Google API
http://api.google.com

[GOOGLE] Google Search Engine
http://www.google.com

[INFORMIT] Comparison of .Net and Java Architectures

[ISC04] Internet Systems Consortium - Domain Survey Information
http://www.isc.org/index.pl/ops/ds/

[JAVA] Java Technology
http://java.sun.com/
<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MSNMES]</td>
<td>Microsoft Network Messenger</td>
<td><a href="http://www.msn.co.uk/messenger/">http://www.msn.co.uk/messenger/</a></td>
</tr>
<tr>
<td>[XMLORG]</td>
<td>Extensible Markup Language (XML) at W3C</td>
<td><a href="http://www.w3.org/XML/">http://www.w3.org/XML/</a></td>
</tr>
</tbody>
</table>
Appendix - Additional Implementation Insights

A.1 Access Pattern UML Diagram
A.2 Access Pattern XML Example

```xml
<AccessPattern ID="47">
  <Page ID="718" Time="08/06/2004 12:33:00" URL="http://www.bbc.co.uk/weather/">
  </Page>
  </Page>
  </Page>
  <Page ID="716" Time="08/06/2004 12:33:30" URL="http://www.bbc.co.uk/weather/">
  </Page>
</AccessPattern>
```
B User Guide

This user guide presents an insight into how to setup and use Damaranke.

The three applications of Damaranke include:
- A proxy server (Proxy Server.exe)
- An intelligence server (Server.exe)
- A client interface (Client.exe)

B.1 Setup

B.1.1. Intelligence Server

1. Load the intelligence server application (Server.exe).

2. Note the port that the intelligence server listens on. This will be needed later.

B.1.2. Proxy Server

1. Load the proxy server application (Proxy Server.exe)

2. Type “addlistener” in the command line
3. Enter the full class name of the Damaranke Listener object. This is:

4. Enter the construction parameters for the listener as:
"host:<PROXYADDRESS>;int:<PROXYPORT>;serveraddress:<SERVERADDRESS>;
serverport:<SERVERPORT>"

where:

<PROXYADDRESS> is the IP address of where the proxy server will reside.
<PROXYPORT> is the port used to listen by the proxy server.
<SERVERADDRESS> is the IP address of where the intelligence server will reside
<SERVERPORT> is the port used to listen by the intelligence server.

5. The settings will be saved. Subsequent launches of the application will automatically
regenerate the last settings used.

B.1.3. Client

1. Launch the client application (Client.exe)

2. Choose a username and password for the system and insert these into the textboxes in addition to the server address and server port. Click Connect.

3. On a successful connection to the intelligence server, you will be greeted by a welcome message.

B.1.4. Browser (Internet Explorer Example)

1. Locate the connection settings used by your browser. In Internet Explorer, these are located on the menu under Tools → Internet Option → Connection (tab).

2. Locate the proxy server settings under LAN settings and configure only the HTTP traffic to use the proxy server. **Note: If the intelligence server is running on the same PC**
as the browser settings being changed, you must also include api.google.com in the exceptions box.

B.2 Tool Overview

B.2.1. Recommended Bookmarks

This tool displays a list of recommended bookmarks.

To add a bookmark to "My Bookmarks", either click the link in the popup window, or select the link from the "Recommended Bookmarks" page and click "Add To My Bookmarks".

B.2.2. My Bookmarks

Use this tool to track a set of bookmarks. The set of Bookmarks can be refreshed by clicking the "Get Bookmarks" button. To go to a bookmark, select the bookmark and click "Go To Bookmark".

B.2.3. Shortcuts

Use this tool to shortcut to pages that you normally visit. Take a shortcut by either clicking the link in the popup window, or select the link from the "Shortcuts" page and click "Go To Shortcut".
B.2.4. Google Recommendations

Use this tool to visit new pages based on your interest. The list of recommended pages can be refreshed by clicking the "Get Google Recommendations" button.

The recommendation summary gives you a brief insight into the selected web page. It includes:

**Rank** – An estimation of the similarity of this page to your interests.

**Title** – A brief description of what the page is likely to contain.

**Link** – The URL of the page.

To go to a recommendation, select the recommendation from the list and click "Go To Recommendation" button.
C Testing

C.1 Technical Access Pattern Test Data

C.1.1. Maximum Forward Path Generation

Test: Maximum Forward Path Generation
MinimumSupport: 1

=== Log ===
ABCDCECBF
ABCD
========

=== Database ===
ABCD
ABCE
ABF
ABCD
=========

=== C1 ===
A: 4
B: 4
C: 3
D: 2
E: 1
F: 1
=========

=== L1 ===
A: 4
B: 4
C: 3
D: 2
E: 1
F: 1
=========

=== C2 ===
AA: 0
AB: 4
AC: 3
AD: 2
AE: 1
AF: 1
BA: 0
BB: 0
### L2 ===

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>4</td>
</tr>
<tr>
<td>AC</td>
<td>3</td>
</tr>
<tr>
<td>AD</td>
<td>2</td>
</tr>
<tr>
<td>AE</td>
<td>1</td>
</tr>
<tr>
<td>AF</td>
<td>1</td>
</tr>
<tr>
<td>BC</td>
<td>3</td>
</tr>
<tr>
<td>BD</td>
<td>2</td>
</tr>
<tr>
<td>BE</td>
<td>1</td>
</tr>
<tr>
<td>BF</td>
<td>1</td>
</tr>
<tr>
<td>CD</td>
<td>2</td>
</tr>
<tr>
<td>CE</td>
<td>1</td>
</tr>
</tbody>
</table>

### C3 ===

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>3</td>
</tr>
<tr>
<td>ABD</td>
<td>2</td>
</tr>
<tr>
<td>ABE</td>
<td>1</td>
</tr>
<tr>
<td>ABF</td>
<td>1</td>
</tr>
<tr>
<td>ACD</td>
<td>2</td>
</tr>
<tr>
<td>ACE</td>
<td>1</td>
</tr>
<tr>
<td>BCD</td>
<td>2</td>
</tr>
<tr>
<td>BCE</td>
<td>1</td>
</tr>
</tbody>
</table>

### L3 ===

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>3</td>
</tr>
<tr>
<td>ABD</td>
<td>2</td>
</tr>
<tr>
<td>ABE</td>
<td>1</td>
</tr>
<tr>
<td>ABF</td>
<td>1</td>
</tr>
<tr>
<td>ACD</td>
<td>2</td>
</tr>
<tr>
<td>Pattern</td>
<td>Frequency</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>ACE</td>
<td>1</td>
</tr>
<tr>
<td>BCD</td>
<td>2</td>
</tr>
<tr>
<td>BCE</td>
<td>1</td>
</tr>
</tbody>
</table>

=== C4 ===

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>2</td>
</tr>
<tr>
<td>ABCE</td>
<td>1</td>
</tr>
</tbody>
</table>

=== L4 ===

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>2</td>
</tr>
<tr>
<td>ABCE</td>
<td>1</td>
</tr>
</tbody>
</table>

### Frequent Patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>AB</td>
<td>4</td>
</tr>
<tr>
<td>ABC</td>
<td>3</td>
</tr>
<tr>
<td>ABCD</td>
<td>2</td>
</tr>
<tr>
<td>ABCE</td>
<td>1</td>
</tr>
<tr>
<td>ABD</td>
<td>2</td>
</tr>
<tr>
<td>ABE</td>
<td>1</td>
</tr>
<tr>
<td>ABF</td>
<td>1</td>
</tr>
<tr>
<td>AC</td>
<td>3</td>
</tr>
<tr>
<td>ACD</td>
<td>2</td>
</tr>
<tr>
<td>ACE</td>
<td>1</td>
</tr>
<tr>
<td>AD</td>
<td>2</td>
</tr>
<tr>
<td>AE</td>
<td>1</td>
</tr>
<tr>
<td>AF</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>BC</td>
<td>3</td>
</tr>
<tr>
<td>BCD</td>
<td>2</td>
</tr>
<tr>
<td>BCE</td>
<td>1</td>
</tr>
<tr>
<td>BD</td>
<td>2</td>
</tr>
<tr>
<td>BE</td>
<td>1</td>
</tr>
<tr>
<td>BF</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>CD</td>
<td>2</td>
</tr>
<tr>
<td>CE</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>
C.1.2. Appropriate Subpath Generation

Test: Appropriate Subpath Generation
MinimumSupport: 1

=== Log ===
ABC
AB
=========

=== Database ===
ABC
AB
=========

=== Subpaths Generated ===
A: 1
B: 1
C: 1
=========

=== C1 ===
A: 2
B: 2
C: 1
=========

=== L1 ===
A: 2
B: 2
C: 1
=========

=== Subpaths Generated ===
AB: 1
AC: 1
BC: 1
=========

=== Subpaths Generated ===
AB: 1
=========

=== C2 ===
AA: 0
AB: 2
AC: 1
BA: 0
BB: 0
BC: 1
CA: 0
CB: 0
CC: 0

=====

=== L2 ===
AB: 2
AC: 1
BC: 1

=====

=== Subpaths Generated ===
ABC: 1

=====

=== C3 ===
ABC: 1

=====

=== L3 ===
ABC: 1

=====

=== Frequent Patterns ===
A: 2
AB: 2
ABC: 1
AC: 1
A: 2
B: 2
BC: 1
C: 1

=====
C.1.3. Susceptibility To Noise

Test: Susceptibility to noise (no noise)
MinimumSupport: 2

=== Log ===
ABCD
ABCD
ABCD

=== Database ===
ABCD
ABCD
ABCD

=== C1 ===
A: 3
B: 3
C: 3
D: 3

=== L1 ===
A: 3
B: 3
C: 3
D: 3

=== C2 ===
AA: 0
AB: 3
AC: 3
AD: 3
BA: 0
BB: 0
BC: 3
BD: 3
CA: 0
CB: 0
CC: 0
CD: 3
DA: 0
DB: 0
DC: 0
DD: 0

=== L2 ===
AB: 3
AC: 3
AD: 3
BC: 3
BD: 3
CD: 3
Appendix - Testing

=========

== C3 ==
ABC: 3
ABD: 3
ACD: 3
BCD: 3
========= 

== L3 ==
ABC: 3
ABD: 3
ACD: 3
BCD: 3
========= 

== C4 ==
ABCD: 3
========= 

== L4 ==
ABCD: 3
========= 

== Frequent Patterns ==
A: 3
AB: 3
ABC: 3
ABCD: 3
ABD: 3
AC: 3
ACD: 3
AD: 3
B: 3
BC: 3
BCD: 3
BD: 3
C: 3
CD: 3
D: 3
========= 

Test: Susceptibility to noise (with noise)
MinimumSupport: 2

== Log ==
ARBSCTD
AUBVCWD
AXBYCZD
========= 

== Database ==
ARBSCTD
AUBVCWD
AXBYCZD
========= 

- 80 -
Appendix - Testing

=== C1 ===
A: 3
B: 3
C: 3
D: 3
R: 1
S: 1
T: 1
U: 1
V: 1
W: 1
X: 1
Y: 1
Z: 1
========

=== L1 ===
A: 3
B: 3
C: 3
D: 3
========

=== C2 ===
AA: 0
AB: 3
AC: 3
AD: 3
BA: 0
BB: 0
BC: 3
BD: 3
CA: 0
CB: 0
CC: 0
CD: 3
DA: 0
DB: 0
DC: 0
DD: 0
========

=== L2 ===
AB: 3
AC: 3
AD: 3
BC: 3
BD: 3
CD: 3
========

=== C3 ===
ABC: 3
ABD: 3
ACD: 3
BCD: 3
========
Appendix - Testing

=== L3 ===
ABC: 3
ABD: 3
ACD: 3
BCD: 3

====

=== C4 ===
ABCD: 3

====

=== L4 ===
ABCD: 3

====

=== Frequent Patterns ===
A: 3
AB: 3
ABC: 3
ABCD: 3
ABD: 3
AC: 3
ACD: 3
AD: 3
B: 3
BC: 3
BCD: 3
BD: 3
C: 3
CD: 3
D: 3

====
C.1.4. Composite Pattern Detection

Test: Composite Pattern Detection
MinimumSupport: 3

=== Log ===
ABC
XYZ
AXBYCZ
XAYBZC
=======

=== Database ===
ABC
XYZ
AXBYCZ
XAYBZC
=======

=== C1 ===
A: 3
B: 3
C: 3
X: 3
Y: 3
Z: 3
=======

=== L1 ===
A: 3
B: 3
C: 3
X: 3
Y: 3
Z: 3
=======

=== C2 ===
AA: 0
AB: 3
AC: 3
AX: 1
AY: 2
AZ: 2
BA: 0
BB: 0
BC: 3
BX: 0
BY: 1
BZ: 2
CA: 0
CB: 0
CC: 0
CX: 0
CY: 0
CZ: 1
XA: 1
XB: 2
Appendix - Testing

<table>
<thead>
<tr>
<th>XC</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX</td>
<td>0</td>
</tr>
<tr>
<td>XY</td>
<td>3</td>
</tr>
<tr>
<td>XZ</td>
<td>3</td>
</tr>
<tr>
<td>YA</td>
<td>0</td>
</tr>
<tr>
<td>YB</td>
<td>1</td>
</tr>
<tr>
<td>YC</td>
<td>2</td>
</tr>
<tr>
<td>YX</td>
<td>0</td>
</tr>
<tr>
<td>YY</td>
<td>0</td>
</tr>
<tr>
<td>YZ</td>
<td>3</td>
</tr>
<tr>
<td>ZA</td>
<td>0</td>
</tr>
<tr>
<td>ZB</td>
<td>0</td>
</tr>
<tr>
<td>ZC</td>
<td>1</td>
</tr>
<tr>
<td>ZX</td>
<td>0</td>
</tr>
<tr>
<td>ZY</td>
<td>0</td>
</tr>
<tr>
<td>ZZ</td>
<td>0</td>
</tr>
</tbody>
</table>

=====

=== L2 ===

<table>
<thead>
<tr>
<th>AB</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>3</td>
</tr>
<tr>
<td>BC</td>
<td>3</td>
</tr>
<tr>
<td>XY</td>
<td>3</td>
</tr>
<tr>
<td>XZ</td>
<td>3</td>
</tr>
<tr>
<td>YZ</td>
<td>3</td>
</tr>
</tbody>
</table>

=====

=== C3 ===

<table>
<thead>
<tr>
<th>ABC</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ</td>
<td>3</td>
</tr>
</tbody>
</table>

=====

=== L3 ===

<table>
<thead>
<tr>
<th>ABC</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ</td>
<td>3</td>
</tr>
</tbody>
</table>

=====

=== Frequent Patterns ===

<table>
<thead>
<tr>
<th>A</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>3</td>
</tr>
<tr>
<td>ABC</td>
<td>3</td>
</tr>
<tr>
<td>AC</td>
<td>3</td>
</tr>
<tr>
<td>BC</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
</tr>
<tr>
<td>X</td>
<td>3</td>
</tr>
<tr>
<td>XY</td>
<td>3</td>
</tr>
<tr>
<td>XYZ</td>
<td>3</td>
</tr>
<tr>
<td>XZ</td>
<td>3</td>
</tr>
<tr>
<td>Y</td>
<td>3</td>
</tr>
<tr>
<td>YZ</td>
<td>3</td>
</tr>
<tr>
<td>Z</td>
<td>3</td>
</tr>
</tbody>
</table>

=====
C.1.5. Overall Test

Test: Overall Test
MinimumSupport: 3

=== Log ===
ABCBC
DEF
ADU8ECVWF
DAEXBFYZ

=== Database ===
ABC
ABC
DEF
ADU8ECVWF
DAEXBFYZ

=== C1 ===
A: 4
B: 4
C: 4
D: 3
E: 3
F: 3
U: 1
V: 1
W: 1
X: 1
Y: 1
Z: 1

=== L1 ===
A: 4
B: 4
C: 4
D: 3
E: 3
F: 3

=== C2 ===
AA: 0
AB: 4
AC: 4
AD: 1
AE: 2
AF: 2
BA: 0
BB: 0
BC: 4
BD: 0
BE: 1
BF: 2
CA: 0
CB: 0
CC: 0
CD: 0
CE: 0
CF: 1
DA: 1
DB: 2
DC: 2
DD: 0
DE: 3
DF: 3
EA: 0
EB: 1
EC: 2
ED: 0
EE: 0
EF: 3
FA: 0
FB: 0
FC: 1
FD: 0
FE: 0
FF: 0

=======

=== L2 ===
AB: 4
AC: 4
BC: 4
DE: 3
DF: 3
EF: 3

=======

=== C3 ===
ABC: 4
DEF: 3

=======

=== L3 ===
ABC: 4
DEF: 3

=======

=== Frequent Patterns ===
A: 4
AB: 4
ABC: 4
AC: 4
B: 4
BC: 4
C: 4
D: 3
DE: 3
DEF: 3
DF: 3
E: 3
EF: 3
F: 3

=======
C.2 Technical User Interest Test Data

C.2.1. Comparison with Single Category of Interest

Single Category Approach

```xml
<Profile ID="40">
  <Category ID="1" Name="">
    <Centroid>
      <Document ID="-1">
        <Term Term="bst" Weight="0.085" />
        <Term Term="classic" Weight="0.043" />
        <Term Term="east" Weight="0.099" />
        <Term Term="fabl" Weight="0.028" />
        <Term Term="forecast" Weight="0.043" />
        <Term Term="live" Weight="0.034" />
        <Term Term="ninja" Weight="0.044" />
        <Term Term="shadow" Weight="0.051" />
        <Term Term="south" Weight="0.052" />
        <Term Term="starter" Weight="0.028" />
        <Term Term="temperatur" Weight="0.048" />
        <Term Term="weather" Weight="0.178" />
        <Term Term="xbox" Weight="0.271" />
      </Document>
    </Centroid>
  </Category>
</Profile>

<Profile ID="716">
  <Category ID="1" Name="">
    <Centroid>
      <Document ID="716">
        <Term Term="east" Weight="0.228169802413295" />
        <Term Term="forecast" Weight="0.256796151435993" />
        <Term Term="weather" Weight="0.515034046150711" />
      </Document>
    </Centroid>
  </Category>
</Profile>

<Profile ID="717">
  <Category ID="1" Name="">
    <Centroid>
      <Document ID="717">
        <Term Term="bst" Weight="0.483912703820121" />
        <Term Term="temperatur" Weight="0.291764260489612" />
        <Term Term="weather" Weight="0.224323035690267" />
      </Document>
    </Centroid>
  </Category>
</Profile>

<Profile ID="728">
  <Category ID="1" Name="">
    <Centroid>
      <Document ID="728">
        <Term Term="east" Weight="0.363325584909888" />
        <Term Term="south" Weight="0.3083554248282" />
        <Term Term="weather" Weight="0.32828887260729" />
      </Document>
    </Centroid>
  </Category>
</Profile>

<Profile ID="781">
  <Category ID="1" Name="">
    <Centroid>
      <Document ID="781">
        <Term Term="fabl" Weight="0.168020803297903" />
        <Term Term="ninja" Weight="0.260025292043389" />
        <Term Term="xbox" Weight="0.571953904658708" />
      </Document>
    </Centroid>
  </Category>
</Profile>

<Profile ID="766">
  <Category ID="1" Name="">
    <Centroid>
      <Document ID="766">
        <Term Term="classic" Weight="0.259730186434759" />
        <Term Term="shadow" Weight="0.312619356770101" />
        <Term Term="xbox" Weight="0.42765045679514" />
      </Document>
    </Centroid>
  </Category>
</Profile>

<Profile ID="522">
  <Category ID="1" Name="">
    <Centroid>
      <Document ID="522">
        <Term Term="live" Weight="0.205332003194336" />
        <Term Term="starter" Weight="0.164897794520336" />
        <Term Term="xbox" Weight="0.629770202285328" />
      </Document>
    </Centroid>
  </Category>
</Profile>
```
Multiple Category Approach

<Profile ID="40">
    <Category ID="1" Name="">
        <Centroid>
            <Document ID="-1">
                <Term Term="bst" Weight="0.161304234606707" />
                <Term Term="east" Weight="0.197165129107728" />
                <Term Term="forecast" Weight="0.0855987171453311" />
                <Term Term="south" Weight="0.102795180827607" />
                <Term Term="temperatur" Weight="0.0972547534965374" />
                <Term Term="weather" Weight="0.355881984816089" />
            </Document>
        </Centroid>
        <Document ID="716">
            <Term Term="east" Weight="0.228169802413295" />
            <Term Term="forecast" Weight="0.256796151435993" />
            <Term Term="weather" Weight="0.515034046150711" />
        </Document>
        <Document ID="717">
            <Term Term="bst" Weight="0.483912703820121" />
            <Term Term="temperatur" Weight="0.291764260489612" />
            <Term Term="weather" Weight="0.224323035690267" />
        </Document>
        <Document ID="728">
            <Term Term="east" Weight="0.363325584909888" />
            <Term Term="south" Weight="0.308385542482822" />
            <Term Term="weather" Weight="0.3282887260729" />
        </Document>
    </Category>
    <Category ID="2" Name="">
        <Centroid>
            <Document ID="-1">
                <Term Term="classic" Weight="0.0865767288115863" />
                <Term Term="fabl" Weight="0.056006934432634" />
                <Term Term="live" Weight="0.068444010647785" />
                <Term Term="ninja" Weight="0.0866750973477963" />
                <Term Term="shadow" Weight="0.1042064522567" />
                <Term Term="starter" Weight="0.0549659315067788" />
                <Term Term="xbox" Weight="0.543124854579725" />
            </Document>
        </Centroid>
        <Document ID="781">
            <Term Term="fabl" Weight="0.168020803297903" />
            <Term Term="ninja" Weight="0.260025292043389" />
            <Term Term="xbox" Weight="0.571953904658708" />
        </Document>
        <Document ID="766">
            <Term Term="classic" Weight="0.2597301864343759" />
            <Term Term="shadow" Weight="0.312619356770101" />
            <Term Term="xbox" Weight="0.42765045679514" />
        </Document>
        <Document ID="522">
            <Term Term="live" Weight="0.205332003194336" />
            <Term Term="starter" Weight="0.164897794520336" />
            <Term Term="xbox" Weight="0.629770202285328" />
        </Document>
    </Category>
</Profile>
C.2.2. Effectiveness of the Gutter

Existing Approach

<Profile ID="41" />
<Category ID="1" Name=""
<Centroid>
  <Document ID="-1">
    <Term Term="bst" Weight="0.161328875841128" />
    <Term Term="east" Weight="0.19689456077976" />
    <Term Term="forecast" Weight="0.085579535615399" />
    <Term Term="south" Weight="0.102735006756065" />
    <Term Term="temperatur" Weight="0.0970301830470948" />
    <Term Term="weather" Weight="0.356431837960554" />
  </Document>
</Centroid>
</Category>
<Category ID="2" Name=""
<Centroid>
  <Document ID="716">
    <Term Term="east" Weight="0.227743662581276" />
    <Term Term="forecast" Weight="0.256738606846197" />
    <Term Term="weather" Weight="0.515517730572525" />
  </Document>
</Centroid>
</Category>
<Category ID="3" Name=""
<Centroid>
  <Document ID="781">
    <Term Term="fabl" Weight="0.166388809907815" />
    <Term Term="ninja" Weight="0.25971191970835" />
    <Term Term="xbox" Weight="0.573899270383835" />
  </Document>
</Centroid>
</Category>

Gutter
New Approach with Gutter

Existing Approach after Extra Document Incorporation
New Approach with Gutter after Extra Document Incorporation
<Term Term="temperatur" Weight="0.0970301830470948" />
<Term Term="weather" Weight="0.356431837960554" />
</Document>
</Centroid>
<Document ID="716">
<Term Term="east" Weight="0.227743662581278" />
<Term Term="forecast" Weight="0.256738606846197" />
<Term Term="weather" Weight="0.515517730572525" />
</Document>
<Document ID="717">
<Term Term="bst" Weight="0.483986627523384" />
<Term Term="temperatur" Weight="0.291090549141284" />
<Term Term="weather" Weight="0.224922823335332" />
</Document>
<Document ID="728">
<Term Term="east" Weight="0.362940019758003" />
<Term Term="south" Weight="0.308205020268194" />
<Term Term="weather" Weight="0.328854959973804" />
</Document>
</Category>
<Category ID="2" Name="">
<Centroid>
<Term Term="fabl" Weight="0.0831944049539074" />
<Term Term="live" Weight="0.103037366787704" />
<Term Term="ninja" Weight="0.129855959854175" />
<Term Term="starter" Weight="0.082249220129618" />
<Term Term="xbox" Weight="0.601663048274596" />
</Document>
</Centroid>
<Document ID="781">
<Term Term="fabl" Weight="0.166388809907815" />
<Term Term="ninja" Weight="0.259711919708335" />
<Term Term="xbox" Weight="0.573899270383835" />
</Document>
</Category>
<Gutter>
<Document ID="111">
<Term Term="financ" Weight="0.195381430558002" />
<Term Term="movi" Weight="0.201668964558706" />
<Term Term="yahoo" Weight="0.602949604883292" />
</Document>
</Gutter>
</Profile>
D Database Stored Procedures

D.1 addBookmark

CREATE PROCEDURE addBookmark
(  
    @bookmarkcategory_id int,
    @ipaddress varchar(50),
    @bookmark_url varchar(255)
) AS

BEGIN TRANSACTION

DECLARE @profile_id int

SELECT @profile_id=Profile_Users.profile_id
FROM Profile_Users
    INNER JOIN CurrentUsers
    ON Profile_Users.UserID = CurrentUsers.UserID
WHERE CurrentUsers.IPAddress=@ipaddress

SELECT *
FROM bookmark
WHERE bookmark_url=@bookmark_url AND profile_id=@profile_id

IF @@rowcount=0
    BEGIN
        insert into bookmark
            (profile_id,bookmarkcategory_id,bookmark_url)
        values (@profile_id,@bookmarkcategory_id,@bookmark_url)
    END

COMMIT TRANSACTION

GO

D.2 Authentication

CREATE PROCEDURE Authentication
(  
    @username varchar(50),
    @password varchar(50),
    @ipaddress varchar(50),
    @result varchar(50) OUTPUT
) AS

BEGIN TRANSACTION

DECLARE @userid int

SELECT @userid=Userid FROM Users WHERE Username=@username AND Password=@password

IF (@@rowcount>0)
    BEGIN
        --Username and password correct so update details

SELECT * FROM CurrentUsers WHERE IPAddress=@ipaddress
IF(@@rowcount>0)
   -- Ipaddress already exists to change the current user
   UPDATE CurrentUsers SET UserID=@userid WHERE IPAddress=@ipaddress
ELSE
   -- New Ipaddress so insert
   INSERT INTO CurrentUsers (IPAddress,UserID) VALUES (@ipaddress,@userid)
   SELECT * FROM CurrentUsers WHERE IPAddress=@ipaddress
   SET @result='usernamepasswordcorrect'
END
ELSE
BEGIN
   -- Check if the username was correct at least
   SELECT @userid=UserID FROM Users WHERE Username=@username
   IF(@@rowcount>0)
      -- Username does exist so password was incorrect
      SET @result='passwordincorrect'
   ELSE
      BEGIN
         -- Username doesn't exist so it is a new user
         INSERT INTO Users (Username,Password)
         VALUES (@username,@password)
         SET @userid=@@identity
         -- Determine the current user now
         SELECT * FROM CurrentUsers WHERE IPAddress=@ipaddress
         IF(@@rowcount>0)
            -- Ipaddress already exists to change the current user
            UPDATE CurrentUsers
            SET UserID=@userid
            WHERE IPAddress=@ipaddress
         ELSE
            -- New Ipaddress so insert
            INSERT INTO CurrentUsers (IPAddress,UserID) VALUES (@ipaddress,@userid)
            -- Create a new profile now too
            INSERT INTO Profile_Users (UserID) VALUES (@userid)
            SET @result='newusercreated'
         END
      END
   END
END
COMMIT TRANSACTION
GO

D.3 getBookmarks

CREATE PROCEDURE getBookmarks
   (   @ipaddress varchar(50)
   )
AS
SELECT Bookmark.URL, Bookmark.BookmarkCategory_ID as BookmarkCategory_ID, BookmarkCategory_Name
FROM Bookmark_Category
INNER JOIN Bookmark
INNER JOIN Profile_Users
INNER JOIN CurrentUsers
INNER JOIN Users
   ON Users.UserID = CurrentUsers.UserID
   ON Profile_Users.UserID = CurrentUsers.UserID
   ON Bookmark.Profile_ID = Profile_Users.PROFILE_ID
   ON Bookmark.BookmarkCategory_ID = Bookmark_Category.BookmarkCategory_ID
WHERE CurrentUsers.IPAddress=@ipaddress
**D.4 getDocumentFrequency**

CREATE PROCEDURE getDocumentFrequency (
    @term varchar(50),
    @DF int OUTPUT
) AS

select @DF=DF
from Term_Document_Frequency
where Term = @term

if (@@rowcount<1)
    SET @DF=-1

GO

**D.5 getIPAddress**

CREATE PROCEDURE getIPAddress
    (@profile_id int)
AS

SELECT CurrentUsers.IPAddress AS IPAddress
FROM Profile_Users
    INNER JOIN CurrentUsers
    INNER JOIN Users
    ON Users.UserID = CurrentUsers.UserID
    ON Profile_Users.UserID = CurrentUsers.UserID
WHERE Profile_Users.Profile_id = @profile_id

GO

**D.6 getLearnPackage**

CREATE PROCEDURE getLearnPackage
    (@logid int)
AS

SELECT Url,Content,CurrentUsers.UserID,Profile_ID,Log.Urlid
FROM Profile_Users
    INNER JOIN CurrentUsers
    INNER JOIN Log
    INNER JOIN Urls
    ON Urls.URLID = Log.URLID
    ON CurrentUsers.IPAddress = Log.IPAddress
    ON Profile_Users.UserID = CurrentUsers.UserID
WHERE Log.LogID = @logid

GO

**D.7 getNumberOfDocuments**

CREATE PROCEDURE getNumberOfDocuments
    (@total int OUTPUT)
AS
select @total=N from Document_Frequency
GO

D.8  getProfileID

CREATE PROCEDURE getProfileID
(  @ipaddress varchar(50)
)  
AS

SELECT Profile_Users.Profile_id AS Profile_id
FROM Profile_Users
INNER JOIN CurrentUsers
INNER JOIN Users
ON Users.UserID = CurrentUsers.UserID
ON Profile_Users.UserID = CurrentUsers.UserID
WHERE CurrentUsers.IPAddress=@ipaddress
GO

D.9  logurl

CREATE PROCEDURE logurl
(  @url varchar(500),
  @content text,
  @ipaddress varchar(50),
  @logid int OUTPUT
)  
AS

--USED DURING PROCEDURE---
DECLARE @urlid int
--------------------------

set @urlid = -1
set @logid = -1

BEGIN TRANSACTION

--Determine if url already exists---
SELECT @urlid=urlid
FROM urls
WHERE url like @url

IF (@urlid > 0)
  --URL already exists
  --Update content (sql only updates if it is different)
  BEGIN
    UPDATE urls
    SET content=@content
    WHERE urlid=@urlid
  END
ELSE
  --New url, so add to db
  BEGIN
    INSERT INTO urls (url,content) VALUES (@url,@content)
    set @urlid = @@identity
  END

--Add appropriate info to the log---
Appendix - Database Stored Procedures

INSERT INTO log (ipaddress,urlid)
VALUES (@ipaddress,@urlid)
SET @logid = @@identity
------------------------------------
COMMIT TRANSACTION
GO

D.10 readAccessPattern

CREATE PROCEDURE readAccessPattern
(  
  @accesspattern_id int
) AS
BEGIN TRANSACTION
SELECT *
FROM Profile_Users
WHERE PROFILE_ID=@accesspattern_id
COMMIT TRANSACTION
GO

D.11 readProfile

CREATE PROCEDURE readProfile
(  
  @profile_id int
) AS
BEGIN TRANSACTION
SELECT *
FROM Profile_Users
WHERE PROFILE_ID=@profile_id
COMMIT TRANSACTION
GO

D.12 removeBookmark

CREATE PROCEDURE removeBookmark
(  
  @ipaddress varchar(50),
  @bookmark_url varchar(255)
) 
AS
DECLARE @profile_id int
BEGIN TRANSACTION
SELECT @profile_id=Profile_Users.profile_id
FROM Profile_Users
INNER JOIN CurrentUsers
INNER JOIN Users
ON Users.UserID = CurrentUsers.UserID
ON Profile_Users.UserID = CurrentUsers.UserID
WHERE CurrentUsers.IPAddress=@ipaddress
DELETE FROM Bookmark WHERE bookmark_url=@bookmark_url AND profile_id=@profile_id

GO
**D.13 saveAccessPattern**

CREATE PROCEDURE saveAccessPattern
(
    @accesspattern_id int,
    @xml text
) AS
BEGIN TRANSACTION
UPDATE Profile_Users
SET ACCESSPATTERN_XML=@xml
WHERE PROFILE_ID=@accesspattern_id
COMMIT TRANSACTION
GO

**D.14 saveProfile**

CREATE PROCEDURE saveProfile
(
    @profile_id int,
    @xml text
) AS
BEGIN TRANSACTION
UPDATE Profile_Users
SET PROFILE_XML=@xml
WHERE PROFILE_ID=@profile_id
COMMIT TRANSACTION
GO

**D.15 updateDF**

CREATE PROCEDURE updateDF
AS
DECLARE @N int
BEGIN TRANSACTION
SELECT @N=N FROM Document_Frequency
UPDATE Document_Frequency
SET N=@N+1
COMMIT TRANSACTION
GO

**D.16 updateTDF**

CREATE PROCEDURE updateTDF
(

@term varchar(50),
@DF int OUTPUT
)
AS
SET @DF = 0
BEGIN TRANSACTION
SELECT @DF=DF FROM Term_Document_Frequency
WHERE Term=@term
IF @@ROWCOUNT = 0
BEGIN
SET @DF=1
INSERT INTO Term_Document_Frequency (Term, DF)
VALUES (@term, @DF)
END
ELSE
BEGIN
SET @DF=@DF+1
UPDATE Term_Document_Frequency
SET DF=@DF
WHERE Term=@term
END
COMMIT TRANSACTION
GO
E  Toolkit API

Full online documentation of the Toolkit API is available at:
http://www.damaranke.co.uk/doc

Optionally you can download the documentation as a single Compiled HTML Help (chm) file at: http://www.damaranke.co.uk/doc/Documentation.chm
Source Code available at:

http://www.damaranke.co.uk