IntelliWeb: Intelligent Web Browsing System
MEng Individual Project Report

The Internet has been the most fundamental change during my lifetime and for hundreds of years. Someone the other day said, "It's the biggest thing since Gutenberg," and then someone else said "No, it's the biggest thing since the invention of writing."

-- Rupert Murdoch

Arjun Panesar
June 2006

Supervisor: Chris Hogger
Second Marker: Krysia Broda

This report, project outsourcing report and all source code available at www.doc.ic.ac.uk/~asp02
Acknowledgements

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires.

-- William Arthur Ward

Firstly, I would like to thank my supervisor, Chris Hogger, for his unrelenting help and support throughout the duration of this project. Chris has been a true source of inspiration, always more than willing to take time out in order to assist me during crucial times in my project.

I must also thank Krysia Broda, for her time during the project. Her advice, especially in regards to relevant reading materials was greatly appreciated.

Thanks must also be given to SFN, the company that assisted me in the testing of my project prototype, and in particular Barry Summers, whose enthusiasm for my project opened my eyes to the limitless possibilities such a project holds. In addition, the time taken out by the BETA testers supplied by SFN was invaluable, especially for the testing and evaluation phase of this project.

I would also like to thank my friends and family; their support and patience has been endless.

Finally, I would like to acknowledge those members of my family who are not with me today. Their experience, wisdom and guidance have helped guide me through life. I thank them for their sacrifices, as without them I would not be half the man I am today.
Abstract

From music lyrics to the fundamentals of quantum mechanics, the Internet has become the modern world’s infinite information resource – with search engines being used by 84% of Internet users to find the data they desire [Fal05]. However, with the exponential growth of the Internet in terms of its size and complexity, the problem is no longer whether one can find a desired article of information. Instead, one is now overloaded with information claiming to be of relevance. The problem has become extracting the information of relevance from that which is irrelevant. With the emergence of Web 2.0, the World Wide Web’s dis-intermediarisation has increased the necessity for major search engines to search intelligently, offering personal recommendations and dynamic amalgamation of content.

With many of the search engines presenting an Internet user with contrasting results for most search terms, the process of extracting information of relevance from them has become increasingly laborious and time consuming.

Envisage a scenario where you were able to give this task to a personal, trained Internet aide that presented only the information that was contextually relevant and localised to your requests; further suggesting only the information to you that you find relevant whilst browsing. Advances in the area of intelligent software agents indicate the prospect of applying these agents to the Internet domain.

This report documents the exploration and implementation of an intelligent web browsing system, designed as a fully functional and effective user application for use by all types of Internet users.
# Table of Contents

Acknowledgements ........................................................................................................................................... i
Abstract ......................................................................................................................................................... ii

1 Introduction ................................................................................................................................................ 1
  1.1 Motivation for the project ...................................................................................................................... 1
  1.2 IntelliWeb Overview ............................................................................................................................. 2
  1.3 Technical Challenges .......................................................................................................................... 3
  1.4 Report Objectives ............................................................................................................................... 5
  1.5 Further note .......................................................................................................................................... 6

2 Background ................................................................................................................................................ 7
  2.1 Intelligent Agents .................................................................................................................................. 7
    2.1.1 What is an intelligent agent? ........................................................................................................... 7
    2.1.2 Intelligent web agents .................................................................................................................... 8
      2.1.2.1 Reconnaissance agents ........................................................................................................... 8
      2.1.2.2 Comparison agents ................................................................................................................ 8
      2.1.2.3 Filter agents ........................................................................................................................... 9
      2.1.2.4 Search agents ........................................................................................................................ 9
    2.1.2.1 Search agents ........................................................................................................................... 9
    2.2 Web mining ....................................................................................................................................... 9
      2.2.1 Web content mining ..................................................................................................................... 9
      2.2.2 Web usage mining ....................................................................................................................... 9
      2.2.3 Web structure mining ............................................................................................................... 10
      2.2.4 User profile mining .................................................................................................................. 10
    2.2 Web mining techniques ..................................................................................................................... 10
      2.3.1 TF-IDF vectors ............................................................................................................................ 10
      2.3.2 Latent Semantic Analysis .......................................................................................................... 10
      2.3.3 Cosine similarity ........................................................................................................................ 11
      2.3.4 Naïve Bayesian classifier .......................................................................................................... 11
      2.3.5 Porter stemmer .......................................................................................................................... 12
      2.3.6 K-Nearest neighbour ............................................................................................................... 12
      2.3.7 Artificial neural networks ........................................................................................................ 12
    2.3 Web mining techniques ..................................................................................................................... 10
      2.4 Current smart web browsing tools ................................................................................................. 13
        2.4.1 Letizia ....................................................................................................................................... 13
        2.4.2 PowerScout ............................................................................................................................. 13
        2.4.3 WebMate .................................................................................................................................. 14
        2.4.4 WebWatcher ........................................................................................................................... 14
        2.4.5 Footprints ................................................................................................................................ 14
        2.4.6 Lira ......................................................................................................................................... 14
        2.4.7 IntelliZap .................................................................................................................................. 15
        2.4.8 Semantic web .......................................................................................................................... 15
        2.4.9 Amalthaea ............................................................................................................................... 15
        2.4.10 Alexa ...................................................................................................................................... 16
        2.4.11 Search engines ......................................................................................................................... 16
        2.4.12 Web directories ...................................................................................................................... 17
      2.4 Current smart web browsing tools ................................................................................................. 13
        2.5 Internet paradigm shifts ............................................................................................................... 18
          2.5.1 Search engine optimisation (SEO) .......................................................................................... 18
          2.5.2 Web 2.0 .................................................................................................................................. 18
          2.5.3 Metasearching ....................................................................................................................... 18
            2.5.3.1 Contextual relevance ........................................................................................................ 18
            2.5.3.2 Local relevance ............................................................................................................... 19
          2.5.4 URL affordance ...................................................................................................................... 19
          2.5.5 Personalisation ........................................................................................................................ 19
          2.5.6 Trust ...................................................................................................................................... 19
        2.5 Internet paradigm shifts ............................................................................................................... 18
      2.6 Critique of existing intelligent web browsing tools .......................................................................... 20
        2.6.1 Techniques to employ ............................................................................................................... 20
          2.6.1.1 Architecture ....................................................................................................................... 20
### 2.6.2 Implementation languages

- Localised recommendations
- Temporal recommendations
- URL affordance
- Personal shortcut recommendations
- Bookmark recommendations
- Personalised news and information
- Interest estimation for immediate personalisation

### 4.2 Techniques to implement

- Autonomous operation
- Document representation
- User interest modelling
- Access pattern modelling
- Metasearch

### 4.3 System architecture

- Real-time operation
- Distributed architecture
- Data storage
- Metasearch architecture
- Implementation language
- Future expansion

### 4.4 System usability

- User notification strategy
- Usability heuristics

### 4.5 Summary of technical decisions

### 5 Design

- System overview
- HTML processing engine
  - Web browser monitoring
- Metasearch engine
- Recommendations engine
- User interface
- System database
- Interest estimation

### 6 Implementation

- HTML processing engine
  - Content extraction
  - Content analysis
  - Use of threading to prevent browser interruption
- Metasearch engine
  - User location

### 3 Specification

- System overview
  - Essential requirements
  - Desired requirements
  - Extensions

### 4 Decisions & Technical Challenges

- System tools
  - Metasearch: contextual and localised marginalisation
  - New document recommendations
    - Localised recommendations
    - Temporal recommendations
    - URL affordance
  - Personal shortcut recommendations
  - Bookmark recommendations
  - Personalised news and information
  - Interest estimation for immediate personalisation

### 2 Summary

- Usability
- Considerations
- Recommendations from existing systems
- Web mining techniques
## Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Number of domain names in existence</td>
<td>1</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Current web browsing</td>
<td>8</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Taxonomy of web mining [Eir03]</td>
<td>9</td>
</tr>
<tr>
<td>Figure 4</td>
<td>User recommendations in Letizia [LET]</td>
<td>13</td>
</tr>
<tr>
<td>Figure 5</td>
<td>WebMate’s proxy server structure [Che98]</td>
<td>14</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Amalthea’s graphical user interface [Mod97]</td>
<td>16</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Comparison of .NET and Java architectures [INF]</td>
<td>24</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Distinguishing features of .NET and Java frameworks [INF]</td>
<td>24</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Representing categories of interest and irrelevant documents</td>
<td>33</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Representing documents in an N dimensional plane, where currently N =3</td>
<td>34</td>
</tr>
<tr>
<td>Figure 11</td>
<td>New user interest modelling</td>
<td>35</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Garbage collection to remove irrelevant documents and topics of lost interest</td>
<td>36</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Representing the benefits of shortcuts</td>
<td>37</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Abstract view on how a metasearch facility would function</td>
<td>39</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Diagram representing the specification and relevance refinement of a web search when a domain is declared</td>
<td>40</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Conceptual system interaction overview of the IntelliWeb system</td>
<td>45</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Processing of a web document by the processing engine</td>
<td>47</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Metasearch engine architecture</td>
<td>48</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Recommendations engine structure</td>
<td>49</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Designed graphical user interface for the IntelliWeb application</td>
<td>50</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Importing bookmarks to estimate user interests</td>
<td>51</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Unified Modelling Language diagram of the HTML processing engine</td>
<td>53</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Example of head and body separation and the use of heading styles</td>
<td>54</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Database schema</td>
<td>58</td>
</tr>
<tr>
<td>Figure 25</td>
<td>IntelliWeb user interface</td>
<td>59</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Bookmark Collector user interface</td>
<td>60</td>
</tr>
<tr>
<td>Figure 27</td>
<td>Use case diagram of the IntelliWeb system</td>
<td>61</td>
</tr>
<tr>
<td>Figure 28</td>
<td>Gantt chart depicting the time management of the IntelliWeb project</td>
<td>64</td>
</tr>
<tr>
<td>Figure 29</td>
<td>Likert scale questionnaire given to testers of the IntelliWeb system</td>
<td>67</td>
</tr>
<tr>
<td>Figure 30</td>
<td>Likert scale ratings</td>
<td>67</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Sample of users and their contextual search results</td>
<td>69</td>
</tr>
<tr>
<td>Figure 32</td>
<td>Technical testing of the bookmark and shortcut functionality</td>
<td>70</td>
</tr>
<tr>
<td>Figure 33</td>
<td>Average Likert scale ratings for usability questionnaire</td>
<td>72</td>
</tr>
<tr>
<td>Figure 34</td>
<td>IntelliWeb user interface and its compliance to Nielsen’s 10 Usability Heuristics</td>
<td>73</td>
</tr>
</tbody>
</table>
### Table of Equations

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TF-IDF vector weight assignment</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Cosine similarity between two vectors</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>Cosine similarity between a query and set of documents</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Naive Bayesian classification</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Euclidean distance</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>TF-IDF vector normalisation [Sal83]</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Pruning of sub-paths to meet minimum support</td>
<td>38</td>
</tr>
<tr>
<td>8</td>
<td>Precision determination</td>
<td>65</td>
</tr>
<tr>
<td>9</td>
<td>Recall determination</td>
<td>65</td>
</tr>
</tbody>
</table>
1 Introduction

The ultimate search engine would basically understand everything in the world, and it would always give you the right thing. And we're a long, long way from that.

-- Larry Page

1.1 Motivation for the project

Few can question the impact the Internet has had on modern life. Availability of information is no longer a concern on the Internet. Yet the continual growth of the Internet has led to a constantly evolving maze of content [Ove03]. The exponential growth of new information sources is indicated by the number of web sites in existence, denoted by figure 1.

![Internet Domain Survey Host Count](Image)

**Figure 1 – Number of domain names in existence**

Given the constantly increasing information overflow of the digital age, the importance of information retrieval has become critical. Web search is today one of the most challenging problems of the Internet [Fin01]. The question has evolved from ‘can I find what I want’ to ‘where is what I want’. This is due to the abundance of inadequate and inappropriate data available on the Internet today.

The main use of the Internet is as an information resource; with 84% of web users using search engines to find the data they require [Fal05]. However, the use of search engines is rather limited, as the search engines do not adapt their search strategies to different users. Due to the diversity of content available on the Internet, a search for the term ‘elephant’ yields first page results which are all to do with insurance or movies (as elephant is both the name of an established insurance company and of a movie released in 2003) [YHO].

Nonetheless, it is still feasible that a user may have been searching for information on elephants as wildlife. In this case, the results displayed to the user would be referred to as noise, as the majority of the pages extracted are irrelevant to a particular user’s interests [Che98]. This is reflected in the statistic that under a quarter of web users say that they always find the information they are looking for [Fal05] and illustrates the lack on contextualisation on the Internet – which is essential to eliminate possible semantic ambiguity.

We are now also witnessing a paradigm-shift in how search engines are being used by Internet users for geographically localised results. For instance, if one were to search for a local estate agent in any of the major search engines, one would be left to sift through the patchy, inconsistent and unhelpful results in order to find any estate agents of interest. Targeted results based on geographic location should be presented to a user, as this has
emerged as the future of search [FOR]. Many Internet users are also looking to more than one search engine for their needs. Instead, querying more specific, smaller search engines to provide them with a better service than is provided by using only one search engine [Wu01]. This is a technique known as metasearching.

Localised results would be of use to any category of Internet user, saving them trawling through endless search results to find specific information or services. Irrefutably, contextual relevance is paramount to any Internet user, and all users would benefit from a system that only produced results of contextual interest.

Coupled with the emergence of Web 2.0, the World Wide Web’s dis-intermediarisation [Dar06] has increased the necessity for search engines to search intelligently, offering personal recommendations and dynamic amalgamation of content. Intelligent search has been described as being based on location, context and personal preferences to give weight to personal recommendations made by a search engine. There is a definitive market for search engines, or new services altogether, to act as brokers, to fill the gap between the services the Internet presents and the various wants and needs of a user.

Furthermore, there is currently nothing in the public domain which monitors a web user’s interests and browsing habits to enhance a user’s browsing experience. Nothing currently remembers who a user is, or what their interests and habits are. It is obvious that information discovery would be enhanced if there were methods of making it personalised [Mou97]. Once a user views an online document or queries a search engine, seldom does their interest in that topic cease, instead persisting for a given duration of time. The nature of a user’s persistence of interest is perhaps one of the most convincing reasons to adopt a set of browsing enhancement tools [Lie95].

Beginners to the Internet would benefit greatly from a system that learned a user’s profile, giving suggestions of pages of specific interest and recommendations; based on the user’s browsing habits. Advanced web users would also benefit from such a tool, which enables them to accurately browse the Internet, saving valuable time. This point is exemplified by the fact that all types of web users visit a small collection of web sites frequently; however around twenty percent of these users have difficulty in retrieving pages they have visited before [Bou01].

By combining current web and profile mining techniques along with innovative approaches to document representation and access pattern recognition, this report aims to document the process of the development of an intelligent web browsing system, referred hereon to as IntelliWeb; to combat the exponential growth of Internet information and its consequential contextual ambiguities.

1.2 IntelliWeb Overview

IntelliWeb is a fully functional client-side intelligent web agent with several focal functionalities. Firstly, it allows users to search the web both contextually and locally, providing only the results relevant to a user’s request. In order to perform a web search, a user must specify a search query. A search is conducted by IntelliWeb through the aggregation of metasearches for a given query. The search results displayed to the user are ranked based on an order of merit, determined by each document’s weighting (formed through a series of heuristics discussed in more depth in 5.1) and each document’s consistency of appearance in each of the individual metasearches.

For contextually relevant results, a user can opt to provide IntelliWeb with an indication of the domain of the search. By doing this, results which are of no contextual relevance are filtered out, resulting in the exhibition of only relevant documents, ranked in an order of merit. In order to be presented geographically localised results, a user simply has to indicate that he or she requires a local search. A user’s location is unobtrusively and autonomously observed through the user’s computer’s location, although this can be refined by the user.
Secondly, IntelliWeb autonomously and automatically constructs user specific browsing models, better known as user interest profiles. It then generates intelligent inferences based on these observed browsing activities, and from these, provides users with a collection of tools that assists them to browse the web sites they frequent in a more effective fashion.

A user’s interest profile is assimilated through the monitoring of a user’s browsing habits. Each web page a user visits is logged and analysed to refine the user’s interest profile to represent the user’s current interests. The user profile is monitored in real-time and is adjusted as the user’s browsing habits vary. Based on the user profile, recommendations of new documents of interest are made to the user, as well as shortcuts to sites that the user frequents when following particular browsing paths. Furthermore, bookmarks are offered to the user of sites that they regularly visit.

Users are able to configure whether they require recommendations, bookmarks and shortcuts to be displayed to them within the main IntelliWeb application, or whether they would like unobtrusive suggestion alerts to be displayed to them.

The adaptive construction of user profiles allows IntelliWeb to additionally provide users with relevant up-to-date news and information in real-time. The re-intermediarisation of information through dynamic personal amalgamation allows the user to be presented only the news and information deemed relevant through their interest profile.

An additional extension to IntelliWeb was also produced, allowing the estimation of a user’s interests before they start using IntelliWeb. Although not prescriptive, this interest estimation allows IntelliWeb to initially determine a user’s interests and offer personalised news instantly, and allows IntelliWeb to build upon an interest profile once the user has started browsing the web to offer documents of interest more precisely. No current item of software which assists a user in their web browsing experience uses any method of establishing an estimate of a user’s interests prior to its use. Instead, the user interest profile is built from scratch once the user begins using the software agent, with an accurate interest profile taking anywhere from hours to weeks of browsing to be formed.

1.3 Technical Challenges

A number of technical challenges had to be addressed during the production of the IntelliWeb tool. First and foremost, in order to construct user specific browsing models, several inherent challenges in web mining had to be addressed [Han00]:

- The World Wide Web is a highly dynamic and evolving information source.
- The complexity of online documents is far greater than traditional text collections.
- The World Wide Web appears to be too big for effective data warehousing.
- Only a small segment of the information available on the World Wide Web is truly relevant or useful to a given Internet user.
- The World Wide Web serves a broad diversity of user communities.

As contextual search is a novel and still relatively new concept, there were many hurdles in its implementation. Firstly, one has to determine whether a document is in actual fact a ‘member’ of a given domain, and if it is (or is not), by what measure. With no known algorithm for this, a heuristic had to be developed to rationally determine whether documents belonged to certain domains. Due to the size of the World Wide Web, making sure the information retrieved was truly useful to the user’s search was essential.

Taking this into account, documents also had to satisfy a more stringent set of web search policies. These were put in place to eliminate the probability of spam web sites and web sites of irrelevance creeping into the search results. This currently occurs with many search engines where web sites ‘cheat’ the index - optimising their web site for specific, targeted
terms illegitimately. This technique is often manipulated by bogus web sites, or sites selling services offered by other companies in the same area as part of an affiliate deal.

These policies had to be implemented without diluting the search term itself. By diluting the search term, through adding supplementary terms to the given search query, web sites optimised for the modified queries would be retrieved. These web sites are less likely to be reputable well-known brands or domain names; as these sites only effectively optimise for their specific brand, service, domain or company name [Lya00].

Due to the complexity of online documents, determining the context and semantics of a document proved a delicate task. Decisions had to be made on how to determine a document’s meaning; whether it was determined by a document’s content, title, META tags or a combination of all of these. This has implications on the subsequent quality of recommendations and performance times. The generation of recommendations of new documents to a user relying solely on content, title and META tags would prove quicker than that which used the document content also. This is due to the fact that this data is much smaller in size in comparison to the actual content (body) of a web page, and hence its parsing is quicker and computationally less expensive.

However, there are many web sites which do not contain META tags, therefore weakening the judgment of the page’s meaning. Also, the use of the actual text body content of a web document helps to substantiate and fine-tune inferences made from the content, title and META tags. In the case of pages without META tags, it is the only way to make a reliable estimation of a page’s meaning. Having chosen to also use the content of documents and their URL’s [Lya00] to judge the meaning of web pages, there lay a further inherent challenge as to how to perform the parsing and analysis of documents without there being any detriment, particularly in responsiveness, to a user's web browsing experience.

The use of a proxy server to monitor web usage was disregarded; due to the difficulty in imposing proxies on many networks, especially networks in places of employment, as such networks commonly operate a prudent firewall [SOA]. Therefore, IntelliWeb itself had to monitor the user's web usage. To do this, the IntelliWeb agent had to interface with the user's web browser in order to monitor their web browsing activities. Due to the quality of service demanded by user's today, this had to be done without slowing down the responsiveness of the user's web browser, making use of the user's computer’s idle time.

Any body of text can have a plethora of meanings. Simply analysing the frequency of terms within a document to estimate the meaning of a document would be insufficient, due to the utilisation of grammar; with the use of prefixes, suffixes, agentives etc, conjunctions and the nature of the English language [Por80].

With the English language, words may have the same underlying meaning although spelt differently. These terms had to be accounted for; else the semantics of the document would become diluted. Hence, methods to determine words of the same meaning had to be implemented, such as the Porter Stemming algorithm. These methods had to be implemented without the speed and efficiency of the monitoring of a web user’s browsing and parsing of web documents being affected. With there often being many connotations to any body of text, extracting the primary meaning of a document proved essential. The use of TF-IDF vectors and research into top-N cropped vectors proved invaluable.

As web users rarely have a single category of interest, it was necessary to model a user interest profile using several categories of interest. With there being many alternatives for the construction of a user's browsing profile, concerns such as computation time, the dynamic nature of human attention and more physical elements such as Internet pop-ups helped to determine the method in which to determine this.

User browsing profiles also had to mirror the wandering nature of human attention. On the web, this can be the pursuing of a new interest, or actions such as following an online advertisement, before continuing with one’s prior interests. Therefore, an element of decay
had to feature in the user’s profile, to gradually remove interests from the user’s current profile which were no longer of interest.

The aesthetic elements of IntelliWeb were also of paramount importance. With the Internet a recurrently used resource, any piece of software used in conjunction with it has to be easy to learn and use. Hence, IntelliWeb also had to meet this criterion to ensure usability. The user interface was critical; hence Nielsen’s 10 Usability Heuristics [USE] were referred to, to ensure the needs of a web user were met. User notifications for recommendations and suggestions also had to be unobtrusive and easily configurable [USE, Lie97].

Finally, the analysis of documents, adaptation of user browsing profiles, and notifications of recommendations and shortcuts to the user all had to occur without effecting the user’s browsing in any way. Hence, the idle time of the computer was used to ensure the user’s browsing experience was not effected.

1.4 Report Objectives

This report comprehensively details the production of the IntelliWeb system and comprises of the following sections:

- **Background**: This section provides research into the topic of intelligent web browsing, including a detailed examination of the underpinning ideas and techniques that could be implemented by an intelligent web system. Furthermore, an examination into the state of the art systems available in the public domain is conducted, with an analysis and evaluation of the current technologies. This section can be found from page 1.

- **Specification**: This section contains the formal specifications of an intelligent web browsing system and the rationale behind the architectural and technical assembly of such a system. This section can be found from page 26.

- **Decisions and Technical Challenges**: Following on from the specification, and to meet the requirements discussed, this chapter identifies and discusses the key decisions made in the scope of the project and the technical challenges involved in their implementation. Approaches examined in the background are evaluated against new techniques and approaches proposed. This section can be found from page 29.

- **Design**: The transition from specification to system design is highlighted. Within this section is contained a detailed description of the system construction, the technical challenges faced, what exactly the system achieves, and how. This section can be found from page 45.

- **Implementation**: The final implementation of the system is discussed, along with the technical problems that had to be overcome during its implementation. This section can be found from page 53.

- **Project Management**: This chapter identifies the project management procedures put in place to ensure the requirements of the IntelliWeb project were satisfied in good stead. This section can be found from page 62.

- **Testing & Evaluation**: In these sections, the test procedures for the final implementation are specified and justified. The results and feedback from the testing procedures are discussed at length and thence form the basis of a critical evaluation of the IntelliWeb tool. These sections can be found from pages 65 and 69 respectively.

- **Conclusion**: This section concludes the report, looking back at the objectives and requirements set out at the beginning of the report, and presents a number of ideas for opportunities of future work. This section can be found from page 77.
Following these sections one can find a bibliography and appendix, with the appendix containing the IntelliWeb user guide and further documentation.

1.5 Further note
The testing of the IntelliWeb system was conducted by a group of fifty BETA testers who had signed up as BETA software testers with SFN. SFN are an e-commerce company based in the Midlands with a twelve year track history and annual turnover of over six million pounds.

Due to the success of the IntelliWeb system testing phase, I was offered a contract from SFN to produce a similar intelligent web browsing agent to IntelliWeb, with several minor alterations.

Further information regarding this can be seen in the conclusion section of this report, and appendix B.
2 Background

This chapter highlights key concepts and techniques required for this project. An introduction to intelligent agents and web mining is followed by an evaluation and discussion of the state of the art.

2.1 Intelligent Agents

If a user’s internet browsing experience is to be enhanced, a tool that encompasses a certain degree of intelligence is required in order to perform tasks ‘intelligently’ for the user’s benefit. For this purpose, the research area of agents proves essential.

2.1.1 What is an intelligent agent?

Russell & Norvig [Rus03] define agents as a system that perceives its environment through sensors and acts upon that environment through effectors. All agents have four elements – performance, environment, actuators and sensors.

An agent can be seen as a mapping between percept sequences and actions. For example, a medical diagnosis system may treat a patient in a hospital as its environment, perceiving symptoms, findings and patient’s answers. It may then act on these through questions, tests and treatments in the goal to have a healthy patient.

There are many different types of agents, which can be classified into categories based on their characteristics. However, agents are able to show multiple characteristics, including:

- **Goal based agents:** These agents make use of the current state description, as well as goal information that describe desirable situations. The agent is able to search and plan to find action sequences that achieve its goal.

- **Utility based agents:** Essentially, a utility function is used to map a state to a real number, which describes the associated degree of happiness. This allows rational decisions in cases where there are several paths to the same goal, allowing an agent to distinguish the paths better than others.

- **Learning agents:** A learning agent is able to operate in an unknown environment and become more competent through experience. Learning techniques remove the need for programming rules, instead allowing the agent to adapt through feedback to changes in its environment in order to modify itself to perform better in the future.

- **Autonomous agent:** The less an agent relies on it’s built in knowledge (as opposed to its own experience) the more autonomous it is. Therefore, an autonomous agent is able to take action independent of the user.

- **Rational agent:** A rational agent is one that does the right thing. This can be represented with a mathematical measure of the agent’s performance, which the agent can seek to maximise.

- **Interface agent:** An interface agent [Lie97] is one that provides an interface to complex system.

An intelligent agent is fundamentally one which is able to display characteristics such as learning, independence, adaptability, inference and to a degree, ingenuity.

For this to occur, an agent must be able to determine sequences of actions autonomously through searching, planning, adapting to changes in its environment and from feedback through its past experiences.
In the realm of intelligent web browsing, an intelligent agent could be applied to:

- Discover contextually relevant documents
- Discover geographically localised products and services of interest to a web user
- Determine a user’s web browsing routines and interests
- Provide feedback in the form of recommendations to aid the web user's internet browsing experience.

2.1.2 Intelligent web agents

Intelligent web agents act on a user’s behalf to make their web browsing experience simpler and more productive. Tasks such as browsing, comparing, searching and filtering data may be conducted by an intelligent agent for the user’s benefit. The agents can be categorised into the following:

2.1.2.1 Reconnaissance agents

Reconnaissance agents are defined as systems that scrutinise a web user’s browsing activities and act as guide, suggesting to the user recommended paths of browsing from their past habits [Lie01]. These agents display characteristics of learning, adaptability and inference; able to infer paths of browsing from observing a web user’s habits and interactions between the system and its user.

Web browsing is treated as an activity which is conducted cooperatively between a computer agent and its human user. The user’s effort varies between active and passive, with the reconnaissance connectivity measured between local and global; where agents trace links locally and where agents use global repositories such as search engines respectively.

Reconnaissance agents can be characterised further by the interfaces they provide. The ‘one input’ interface involves users inputting specific queries to search engines based on the topic and information they require. The ‘zero input’ interface analyses the user’s past browsing behaviours to infer future paths of interest.

2.1.2.2 Comparison agents

Comparison agents supply a web user with appropriate results from content mining on the content of web pages. A comparison agent searches web sites using heuristics in order to supply the user with relevant related data. Often comparison agents are used to provide the lowest priced product or service. An example is Froogle [FRO], which provides the price of products from a plethora of web sites.
2.1.2.3 Filter agents
Filter agents effectively ‘sieve’ data in order to provide a user with relevant information. Usually a user is able to specify a query in which the data is then sorted according to.

2.1.2.4 Search agents
Search agents act as aides which grant a web user the benefit of not having to find the perfect search query in order to find the information they are looking for. These agents often query popular search engines with the data recorded from the user profile to provide the user with relevant search results.

2.2 Web mining
Web mining [Coo97] is the discovery and analysis of useful information from the World Wide Web. Web mining uses data mining techniques to ascertain interesting, useful patterns and implicit information from artefacts or activity related to the World Wide Web.

Within web mining reside four distinct domains which shall be discussed in further detail: web content and web usage mining which were proposed in 1997 [Coo97]; and web structure and user profile mining. The latter domains was introduced [Eir03] as web sites began to personalise their content for individual users.

Figure 3 depicts the evolved definition of web mining as it stands currently.

![Figure 3 - Taxonomy of web mining [Eir03]](image)

2.2.1 Web content mining
Web content mining [Fan04] is defined as the search for new information from web data. Typically, more systematic data is retrieved for a desired topic, specified in most instances by a user. Web content mining goes beyond simple keyword extraction due to the fact that web pages and documents present no machine readable semantics. Web content mining can be split into two separate approaches.

An agent based approach directly mines the contents of documents on the World Wide Web. This is known as information retrieval. A database approach to web mining improves on the content search of other tools such as search engines with regard to the database it uses.

2.2.2 Web usage mining
Web usage mining refers to the analysis of user interactions in order to discover user access patterns and predict users’ behaviours. This is most often deciphered from web server logs. A web server records data regarding the requests made by the user to the server for resources in files named web access logs. These logs can thus be analysed to determine user behavioral patterns.
However it has been proposed [Fen03] that web server logs are often inadequate to determine user behaviours and therefore a client-side monitoring system should be implemented to collect more data from the user side in order to determine more sound user behavioral habits. Click stream data, cookies and user queries have been proposed as other methods to discover a user’s online habits.

2.2.3 Web structure mining
The web consists [Fan03] of both page contents and hyperlinks. Web structure mining analyses the internal structure of an online document to discover the link structure of the hyperlinks at the inter-document level. Hyperlinks to a document can be used to determine the popularity of a document and specific authorities in particular topics [Cha99]. Hyperlinks from a document can be used to determine online hierarchies and more importantly give an indication as to the richness of a document. Web content mining information can be used in conjunction with web structure information to categorise and analyse online hyperlinked communities

2.2.4 User profile mining
Users often specify details such as demographic information (such as name, age, location) and customer profile information (such as interests, preferences, etc) when using web sites. As a collective, these user profiles can be mined in order to determine the extent to which individual customisation can be applied to community customisation. User profile mining is often used by e-commerce companies and online organisations in order to operate community-based systems.

2.3 Web mining techniques
In order to fully understand approaches to smart web browsing and analyse current smart web browsing systems, it is essential to understand the techniques available for use by smart web browsing tools to perform web mining.

2.3.1 TF-IDF vectors
A document can be represented as a high-dimensional vector in the space of words [Sal83]. Each entry in the vector corresponds to a different term within the documents and the number of its occurrences. The TF-IDF vector scheme assigns each term within the document a weight using the following formula:

$$W_{td} = f_{td} \cdot \log \left( \frac{D}{ND_t} \right)$$

Equation 1 – TF-IDF vector weight assignment

Each term’s weight ($W_{td}$) is calculated by multiplying the frequency of the term ($f_{td}$) by the log of the total number of documents ($D$) divided by the number of documents the term occurs at least once in ($ND_t$). The ordering of the terms may not necessarily be maintained.

Weights for the terms gathered can then be used to determine documents with high frequencies of the specific term within them. A collection of TF-IDF vectors could be used to represent a web user’s interest.

2.3.2 Latent Semantic Analysis
Where there is a corpus of a large number of documents, for each document $d$, the dimension of the vector representing each document can typically exceed several thousand. Latent semantic analysis relies on the fact that intuitively, terms in documents may often be related.

For example, if a document $d$ contains the term sea, it will often contain the word beach. Equivalently, if the vector representing $d$ has a non-zero component in the entry for sea, it will
also have a non-zero component in the beach component. If this kind of structure can be detected, relationships between words can be automatically discovered from the data.

The word-document is represented by a matrix $A$, which is decomposed using Singular Value Decomposition in order to give the strength of the most significant correlations and their directions. The decomposition of $A$ allows one to discover the semantics of the document through term correlations and their significance within the document.

The latent semantic analysis method could be applied in order to determine the context of a variety of documents in order to present a web user with results that were contextually relevant.

2.3.3 Cosine similarity
The cosine similarity is a metric that measures the similarity between two vectors. Therefore, this could either be used to define the similarity between two documents, or a document and a query [Rus05]. The cosine similarity between two vectors can be defined as:

$$\text{Sim}(u, v) = \frac{u \cdot v}{||u|| \cdot ||v||}$$

Equation 2 – Cosine similarity between two vectors

The similarity between two vectors is calculated as the inner product of the vectors, divided by the product of the lengths of the vectors in question. Intuitively, the greater the angle between two documents, the less similar they are. This holds for vectors in any $N$ dimensional space.

Furthermore, the TF-IDF vector scheme can be integrated with the cosine similarity metric to determine the similarity between a search query and a plethora of documents [Rus04]:

$$\text{Sim}(q, d) = \frac{\sum_{t \in q \cap d} W_{td} \cdot W_{rq}}{||d|| \cdot ||q||}$$

Equation 3 – Cosine similarity between a query and set of documents

The similarity between a query $q$ and document $d$ is calculated as the product of the TF-IDF weights for the term in both the document and query summed over all the terms in both the query and document; divided by the product of the length of the document and length of the query.

Practically however, calculating $\text{Sim}(q, d)$ would prove computationally expensive as the number of documents used grows [Rus04].

2.3.4 Naïve Bayesian classifier
The Bayesian classifier is based on the Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

Given a specified threshold, this method can be used to classify probabilistically whether a vector representing a document is of interest to a user. Given an attribute $d$, we can calculate whether the example belongs in the class $C$ with the following formula [Fia02]:

...
\[
P(C = c \mid D = d) = \arg \max_c \frac{P(D = d \mid C = c)P(C = c)}{P(D = d)}
\]

Equation 4 – Naïve Bayesian classification

A Bayesian approach uses a prior distribution \( P(C) \) and a likelihood \( P(D \mid C) \). Both of these probabilities can be estimated from training data. In order to calculate which class an example belongs to, the probability of being in each class is calculated. The class assigned to an example is the class that produced the highest probability for the example.

As this algorithm calculates one value for each document, intuitively, interesting documents should receive higher values whereas less interesting documents would receive lower values. An interest threshold can then be used to determine whether a document is of interest.

2.3.5 Porter stemmer

The Porter stemmer algorithm [Por80] is a useful method to improve the effectiveness of information retrieval. The algorithm works on the principle that many words in English share a common root. Stemming works on suffixes, and removes common morphological and inflexional endings from words. Therefore stemming allows one to reduce similar words into a common root form. For example:

“…I felt troubled by the fact that my best friend was in trouble. Not only that, but the issues I had dealt with yesterday were still troubling me…”

The words troubled, trouble and troubling all share the common root trouble. Therefore, according to the Porter stemmer algorithm, instead of counting all three words once, the stem trouble is counted thrice instead.

The benefit of stemming is that common words can be clustered under a common stem, in order to provide a more accurate statistical representation of the number of occurrences of a certain word. However, a drawback of stemming is that the semantic meaning of the word may be lost.

2.3.6 K-Nearest neighbour

The \( k \)-nearest neighbour method classifies an unknown object \( O \) with the label of the majority of the \( k \) nearest neighbours. A neighbour is deemed nearest if it has the smallest distance in feature space.

Each object represents a point in \( N \)-dimensional space. The classification of a new sample is based on learning by analogy. The distance between an unseen object and its neighbour is calculated using the Euclidian distance between \( A = (a_1...a_n) \) and \( B = (b_1...b_n) \) as follows:

\[
d(A, B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}
\]

Equation 5 – Euclidean distance

A disadvantage of this method is its large computing power requirement, since for classifying an object its distance to all the objects in the learning set has to be calculated.

2.3.7 Artificial neural networks

Artificial neural networks [Smi96] are a different paradigm for computing, based on the parallel architecture of animal brains. Neural networks are well equipped for data mining tasks due to their ability to model multi-dimensional data and their efficiency at finding hidden patterns amongst data.
Artificial neural networks are trained with learning data, and learn by examples. The benefits of artificial neural networks are in that they are able to generate models of data based on their training data alone.

However a problem in this lies that the artificial neural network may overfit the data, meaning that it simply learns the results for the data and not the general pattern which emerges from the data. Furthermore, artificial neural networks do not present a clear interface in order for a user to understand their findings.

2.4 Current smart web browsing tools

2.4.1 Letizia

Letizia [Lie97] is an autonomous interface agent for web browsing that treats search through the web space as a continuous, cooperative venture between the user and a computer search agent. Letizia offers continuous related recommendations to other web pages/sites to a user through analysing the user’s profile and browsing activity.

Lieberman does not disclose how user profiles and learned, however documents are treated as a set of keywords and topics, with weights assigned to keywords to represent the significant subjects of a page.

Letizia compiles a user’s profile in real-time, by recording the visited URLs and pages chosen by the user. Furthermore, as there is no user input, Letizia relies on implementing a breadth first search strategy using the user’s position as the root, within the boundary of nearby pages using content and context based heuristics to discover recommendations. Recommendations are made in a preference ordered list.

Letizia’s goal is to provide a web user with ‘intelligent assistance’ and is a prime example of a reconnaissance agent. According to Lieberman, the goal is not to retrieve the ‘best answer’, as is with a search engine, but to make the best use of the most limited and valuable resource - the attention of the user.

Should a user ever be unsure of where to browse to next, Letizia is always able to give recommendations in a non intrusive and simple to follow manner. Letizia is implemented in Macintosh Common Lisp and runs in parallel with the Netscape internet browser.

2.4.2 PowerScout

PowerScout [Lie01] is similar to Letizia, and is another Lieberman innovation. Keywords are extracted and weighted similarly to Letizia, through methods such as word position, utilisation
and frequency. However, PowerScout then queries a search engine to determine new recommendations that correlate to the user’s profile of interests.

As a search engine does not always present the most relevant page as its first match, PowerScout presents a dynamically changing list of recommendations. As with Letizia, PowerScout is implemented with Netscape internet browser.

### 2.4.3 WebMate

WebMate [Che98] consists of a standalone proxy that monitors a user’s actions in order to provide information for learning and search refinement. The user profile is learned continuously, using TF-IDF vectors to represent documents, using the cosine vector function to determine similarity.

Users can also explicitly specify pages they like; which allows the system to learn the user’s interests. Users are also able to communicate with the system through feedback to improve recommendations.

![WebMate's proxy server structure](image)

Figure 5 – WebMate’s proxy server structure [Che98]

An applet controller provides the user-proxy server interface. WebMate also presents the user with a personalised newspaper based on the user’s profile and results obtained from search engines and user-specified URLs.

### 2.4.4 WebWatcher

WebWatcher [Arm97] is a goal-orientated information search agent. A web user must invoke WebWatcher by following a link to its page and specify what information is sought. As the user traverses the web, WebWatcher monitors the user’s behaviour and constructs new training examples in order to return link suggestions that may lead to the user’s goal.

Each hyperlink encountered is evaluated using a utility function which assigns it a value between 0 and 1; based on the page, goal, user and link. Links with higher utility values over a given threshold are recommended. The WebWatcher system makes use of TF-IDF vectors, Boolean features, statistical prediction and cosine vector similarity.

### 2.4.5 Footprints

Footprints [Wex97] aims to assist users in the browsing of web sites by visualising the paths, or footprints, taken by previous users who have browsed the same web site. A new user to a site is then able to use these footprints to distinguish the more interesting pages to visit. A back end system is used to process a web site’s web logs, and a front end Java applet is used to read the output data files in order to create a customised site map for each user.

The results of statistical analysis performed on the output data can be displayed graphically through intensity graphs of linked nodes in order to envisage how frequently available paths are used.

### 2.4.6 Lira

Lira is an autonomous agent that searches the internet on behalf of a user for relevant web pages. Lira represents documents as vectors using the Vector Space Model [Sal83] and
introduces Porter stemming, so that terms are represented by their stems to increase context succinctness. Weights are then given to the document terms using an extension of the TF-IDF which introduces normalisation for a given document length:

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

\[u_i: \text{ weight of term } i\]
\[n: \text{ number of documents in corpus}\]
\[tf(i): \text{ document frequency of term } i\]
\[tf_{max}: \text{ maximum term frequency over all words in document } T\]
\[f(i): \text{ document frequency of term } i\]

Equation 6 – TF-IDF vector normalisation [Sal83]

Document relevance is calculated by computing the dot product of vectors \(V\) and \(M\), where \(V\) represents the document vector and \(M\) the user profile vector. Feedback from the user on the usefulness of presented pages is also used to update the user profile.

2.4.7 IntelliZap
IntelliZap [Fin01] is an agent which proposes a new paradigm for performing search in context. Instead of a user specifying a query, search is initiated from a text query marked in a document, and is guided by the words surrounding the marked query – i.e. the context. The context guided information retrieval process involves semantic keyword extraction and clustering to generate new, augmented queries.

Keywords are represented in the Vector Space Model, extracted from the captured text and context. The query is then classified to a predefined domain and sent to a small set of specialised search engines along with highly weighted terms from the META tags. The results from the search engines are then re-ranked using a proprietary ranking algorithm, based on word frequency, link analysis, popularity data and priority listing. In testing, IntelliZap outranked all of the leading search engines.

Information regarding IntelliZap was obtained from [Fin01], submitted to WWW10 in Hong Kong. However on further research, absolutely no evidence of an IntelliZap system was found, with the specified paper the only existing reference found to IntelliZap.

2.4.8 Semantic web
The Semantic Web is a project which seeks to give meaning to contents of documents on the web, in a manner understandable to machines. The aim is that computers will be able to understand the semantic data by following links to key term definitions and rules for reasoning about the terms. The Semantic Web is comprised of the standards and tools of XML, RDF and OWL.

XML provides syntax for structured documents to be annotated for machine analysis. RDF is used to express a data model for referring to objects and their relations.

Finally, OWL adds more vocabulary for expressing ontologies, describing properties and classes. Inference rules can also be applied to ontologies to add more weight to definitions.

2.4.9 Amalthaea
Amalthaea [Mou97] is an evolving, multi agent system. Amalthaea consists of two types of agents; information filtering agents which adapt to the user’s interest, and information discovery agents which monitor and adapt to the online information sources.

Moukas believes that users should rely on search engines, and thus Amalthaea uses their query results as the root for its agents. Amalthaea sends several filtering agents to query search engines and filters the results using an adapted TF-IDF measure.
2.4.10 Alexa

Alexa [ALE] is a system which gathers web information by continually crawling all publicly available web sites to create a series of snapshots of the web. The Alexa crawl collects data which includes site information, incoming and outgoing links and traffic information. This is used to give traffic rankings, pictures of sites, links pointing to sites and a list of related sites that are similar to a given web page.

Alexa users contribute to the data collected also, as users with the Alexa toolbar are able to contribute web usage information which is used to improve traffic rankings and related links. The only disadvantage of Alexa is that not all web sites and crawled, as documents with a certain line of code prevent the pages from being indexed.

2.4.11 Search engines

There are a number of search engines [GOO, YHO, MSN, ALT] which index the World Wide Web and retrieve web based documents based on a user query. Search engines have indices that are built up by robots or crawlers. Search engines use programs known as robots, crawlers and/or spiders that have the following functions: to locate web pages, to read the contents of the web pages and report its findings back to the search engine's indices or databases.

When users use a search engine to locate web sites that are relevant to the keyword search, they are searching the search engine's index.

A search engine with a larger and more up-to-date index is a better representation of the information available in the web. Search engine users can choose whether to search all of the documents in the web, or a subset of these documents which can be narrowed down by country searches.

Search engines distribute their index store over a plethora of storage devices due to the enormous size of the index. Each set of storage devices which holds a complete index is known as a databank. Each search engine will own a number of databanks, each with a more recent index of the Internet. Each databank is updated in turn with a more recent index of the web. Hence, depending on the databank queried, the search results returned to a user for a search term can vary, especially for highly competitive keywords where websites are indexed often.

Search engines [Fal05] are used by 84% of Internet users to find the data they require. However, a common problem is that the information returned is irrelevant to what the user is anticipating. There are several reasons for this:

a. **Poor Choice of Keywords:** Keywords used to query the search engine are a poor choice with regards to the documents returned. This is often due to a lack of experience. Typically half of web search users modify their search query once they have seen the results retrieved [Fal05].
b. **Poor Search Engine Cache:** Specific search engine has a poor set of indexes in cache. This can occur when web sites enter a piece of code within a web document which prevents the document from being indexed. Furthermore, the results returned depend on the search engine databank queried.

c. **Incorrect Context:** Documents returned are technically correct for the query; however due to semantic ambiguity irrelevant results have been retrieved. This can be illustrated by the following example for when a user searches for the term *elephant*. The term *elephant* can refer to several contexts: wildlife, finance (as *elephant* is a car insurance company), or even travel (*Elephant & Castle* is a tube stop).

Research from iProspect [BBC] has shown that search engine users expect to find exactly what they want on the first page of results, with only 10% venturing further than the first page. Furthermore, the significance to companies for achieving a first page ranking is exemplified by the fact that two thirds of users linked first page results to top brands.

The top three search engines ranked by number of users are currently Google, MSN and Yahoo [Fal05, SEA]. Google has the largest index of results, ranked by general popularity with cached copies of a site when a site is unavailable. Google indexes web pages using stemming and also indexes other file types including .PPT, .PS and .WPD files.

MSN has a query building search facility and automatically local searches on a user’s behalf. However, it does not perform stemming and only indexes a subset of the document extensions Google can. Finally, Yahoo! is similar to Google but automatically provides local searches and gives preference to generic URLs and includes pay for inclusion sites. Each search engine uses a distinct algorithm to rank the documents on the web.

However, these algorithms are kept fairly secret due to ‘spam’ web sites entering the search rankings by ‘cheating’ the index - optimising a site for specific terms illegitimately. This technique is often manipulated by bogus web sites, or sites selling services offered by other companies in the same area as part of an affiliate deal. This often involves strategic placement and repetition of keyword terms. Web sites that conduct this technique are rarely present in the search rankings for longer than 48 hours.

To combat this sort of activity, algorithms now use a variety of factors to rank web pages, including reciprocal links, keyword density and content analysis. Nevertheless, spam web sites still occasionally manage to filter through into search engine results. The search engines described above all penalise a web document (give it a lower ranking) [SER] if for a given keyword, the keyword accounts for more than 5% of the total terms in the document. However, even slight changes to this can affect rankings. The threshold used to be set at 10%, however according to search engine experts [SER] this allowed spam web sites to filter through into their results. Latest search engine research states that this percentage needs to be lowered to 4%.

Search engines may be queried more specifically, using “term” to search for documents with an identical match to the term and [term] for broad matches to a given term. There has been a shift in the expectations users have about the relevance and types of results search engines provide. In addition, the way in which web users navigate the web has also changed. This is discussed in more detail in section 2.5.

**2.4.12 Web directories**

Search engines have indices that are built up by robots or crawlers; whereas web directories build up their indices through human editors. The Open Directory Project [ODP] is a web directory. Directories use human editors to review sites that are submitted for submission to the directory. Directories, unlike search engines, use a hierarchical tree structure to organize their database.
Another common distinction is that a directory tends to list web sites (root directory of a site or homepage) whereas a search engine will list web pages (individual pages of a web site). Due to the manual process of adding sites to a directory, directories are guaranteed to have relevant web sites listed under its categories, and the scenario of ‘spamming’ a web site into a directory listing is impossible due to the human editorial technique used.

2.5 Internet paradigm shifts

There is now a generation which takes for granted the Internet as a doorway to an infinite library. As with any technology evolving with a generation, as the technology is exploited, so increases the demands upon what is expected from it.

2.5.1 Search engine optimisation (SEO)

With the sheer numbers using search engines daily, being on the first page of a search engine’s results means exposure to millions. Most web sites and online businesses now optimise their web sites for the world’s search engines. This involves making the web site more ‘search engine friendly’ for the search term(s) the web site wants to rank highly for, so that the web site is highly ranked. This is done in a number of ways, and includes methods such as correct HTML mark-up, lack of dead links, high link popularity and a variety of other methods.

As a result, sites that are optimising for the same term may all feature very highly in many of the search engines leading to many contextually irrelevant results for a term. Furthermore, poor content web sites are beginning to be able to feature highly in search engine results by exploiting SEO techniques leading to irrelevant search results.

2.5.2 Web 2.0

The web itself is changing. With the emergence of Web 2.0, the World Wide Web’s disintermediarisation [Dar06] has increased the necessity for search engines to search intelligently, offering personal recommendations and dynamic amalgamation of content. The Web 2.0 is leading to the belief that intelligent search will eventually be based on location, context and personal preferences to give weight to personal recommendations made by a search engine.

However, many users, instead of relying on search engines, are currently using a number of web sites for their various differing needs rather than relying on search engines for all of their purposes. There is a definitive market for search engines, or new services altogether, to act as brokers, to fill the gap between the services the Internet offers and the various wants and needs of a user.

2.5.3 Metasearching

Metasearching refers to the use of many smaller, more specific (local) search engines to find the information required by a web user. Two factors [Men99] are contributing to the increased use of metasearch:

2.5.3.1 Contextual relevance

As discussed in 2.4.11c, a search term that is technically correct may still retrieve irrelevant results in a major search engine. Due to this, web users are beginning to turn to special purpose search engines which cater for the specific domain. More advanced web users may narrow down their search terms by using extra words in their search queries, usually terms regarding the broader topic of conversation [Fal05].

With the increasing presence of SEO, it has become common practice for web sites to specify their sphere of influence [Fal05] within TITLE, META DESCRIPTION and KEYWORD tags.
2.5.3.2 Local relevance
Web users now also use the Internet as a directory service, searching for local products and services on the Internet. Due to the high degree of SEO used by online businesses, finding a local product or service is extremely frustrating.

Web users are now beginning to use a plethora of smaller, geographic and domain specific local search engines; PPC engines [MIV] and directories [YEL] rather than commercial search engines to find the data they require. Search engine users also expect results based on their country when querying search engines [Fal05] rather than the more general results retrieved from searching the entire web.

2.5.4 URL affordance
As web users become increasingly net-savvy, there is an increasing awareness of bogus links. That is, links that do not lead to the site described and/or lead to spam sites. As a result, URL affordance [Lya00] plays a large part in whether a web user is likely to visit a link. Web users are more likely to follow URLs if:

a. The domain of the URL retrieved is a generic domain name or established company name.

b. The URL address is easy-to-type, correctly spelt, single case, identifies relevant keywords and gives an indication of the website structure.

For example, when searching for mortgages, a web user is more likely to have trust in web sites with URLs such as www.mortgages.co.uk and www.abbey.com, than web sites with URLs such as www.why-not-click-here-for-your-money.com/mortGaGes/002949950.htm.

Furthermore, over half [Fal05] of search engine users click on only the top 5 results indicating that users believe that anything after this may prove irrelevant, or these may be the immediate results seen by the user on screen.

2.5.5 Personalisation
The increased demand from users for information of contextual and localised relevance has, along with the evolution of Web 2.0, led to personalised news and information becoming easily available to a user [RSS]. This is most often done through the use of RSS feeds.

RSS is a format for syndicating news and the content of news-like sites and news-oriented community sites. Documents that can be broken down into discrete items can be syndicated via RSS. Once information is in RSS format, an RSS-aware program can check RSS feeds for changes and react in appropriate ways. RSS-aware programs known as news aggregators are popular online, both checking RSS feeds and displaying new items from each of them.

Presently a user specifies the RSS feeds he/she wishes to view. However, not all of the information presented to the user may be of relevance. Therefore, through the use of a user’s profile and current interests, up-to-date information (most often news) can be displayed to the user from the RSS feeds added to their profile.

2.5.6 Trust
Along with URL affordance and the emergence of Web 2.0 comes the notion of trust [Dar06]. Users are more likely to frequent a particular web site, or use a particular web service, if the site or service they are potentially going to use is trusted by other users.

The mechanism by which one should measure trust in a web site or service is a debatable subject – with many notions put forward. However, feedback from users has proved the most popular form of quantifying trust. This method is illustrated by web sites such as eBay [EBA], which allow users to offer feedback on other users. This then resembles the amount of confidence the online society has on a particular user.
This method has proved extremely successful, with other websites such as Amazon [AMA] and Yahoo! [YHO] following suit.

### 2.6 Critique of existing intelligent web browsing tools

With the existing approaches to intelligent web browsing all contributing to research in this domain, it is essential to analyse each system in order to ensure the proposed system builds upon the successes of existing systems, whilst avoiding the unfavourable aspects of others.

A variety of tools are in existence which attempt to aid an Internet user by providing assistance in the form of browsing, searching and filtering the information available on the web. However, there is currently no product which incorporates a combination of these features in order to tackle the tasks a web user must face. The functionalities implemented in existing systems are discussed below.

#### 2.6.1 Techniques to employ

The methods used by each tool to approach the subject of smart web browsing are analysed below, followed by recommendations which could be implemented in the final proposed system.

##### 2.6.1.1 Architecture

Existing systems have used a variety of approaches to understand a web user’s behaviour; real-time and non real-time tracking.

WebMate uses the real-time, proxy-server approach and logs all HTTP requests to track web usage. This approach creates a client-server relationship between the user and web agent. Although this method offers architectural flexibility, it is frowned upon by a large majority of web users who disagree with the surrendering of trust and personal information [Som99]. Furthermore, such architectures are difficult to impose on many networks, especially networks in places of employment; as such networks commonly operate a prudent firewall. This explicitly denies all types of web proxies and services unless otherwise stated in the firewall configuration. Hence the use of a proxy in order to monitor a user’s web usage would be problematic. Furthermore, users would also need to understand how to configure the proxy for use, which could cause further problems for a potential user. Another approach is that of modifying a user’s browser application.

Letizia uses a modified Netscape browser, and WebWatcher a modified Mosaic browser. However, the main disadvantage of this technique is that the web browsers these tools were created for are no longer popular among the majority today. In addition, this technique also relies on the user owning the web browsing application required. The most popular web browser is currently Internet Explorer with at least 75.5% of web users [BRO], so any application that was to use a modified browser would be best targeted at Internet Explorer users. However this does narrow down the potential scope of the system.

Non real-time tracking uses data from the logs of web sites, which determines where web users have been browsing. This data can range from 24 hours to several months in age. Demonstrated in Footprints, the information collected can be used to help new users navigate a site. However only statistical inferences can be made for such data, as users are anonymous to the web site and there is no user specific feedback. However in order to provide a user with instant recommendations and assistance, it appears that the better approach is through real-time fly on the wall monitoring as discussed formerly. Lieberman [Lie95] also suggests when to incorporate the learning in an intelligent architecture. Web users often pause in their browsing sessions, for example to read documents. Lieberman uses this idle time to search and evaluate, improving the user’s interest profile.

Communication between components of an intelligent web agent would prove most successful if communicating using SOAP on a TCP transport layer. The reason for this is that with firewalls becoming commonplace and more prudent, most prevent the use of all ports.
except 80, the default port for HTTP. System administrators fear opening other ports could pose a potential security problem. Hence the use of SOAP under port 80 ensures that messages will pass through the firewall.

2.6.1.2 Web mining techniques

Web mining can be applied to profile, content and usage mining in this scenario:

Web Profile Mining: Web profile mining allows the customisation of web documents to users’ desires. However, none of the systems analysed take into account the geographic location of the user, hence recommendations provided to the user cannot be specifically localised to a user’s requirements. In order to provide users with geographically local recommendations, if a user was to specify his or her city, country and/or postcode, documents could be recommended that were local and thus more likely to interest the user.

In addition, if the proposed system was to ask for the user’s location, it would allow a naïve to intermediate user the benefit of having local search engine results retrieved with minimal effort. Most advanced users would know that adding extra terms to a search query would aid localised result retrieval. No current tool explicitly allows the search for local services or information; hence this would be a step to solving the contextual and local irrelevance frustration users’ face.

Web Content Mining: Currently no tool allows the explicit search of the web, instead offering recommendations of interest. IntelliZap allows a user to search, however it is bounded by the terms in the web document the user is viewing. The user has to highlight the term they would like to search for in the document. The keyword selected, terms close to the selected keyword and META tags are used to define a domain for the document; which is used along with the keyword to generate contextually relevant results for the user to be recommended.

It would prove useful to provide a user with the ability to search for what they desire, implementing similar contextual, and localised searching. In order to solve the problem of contextual search, if the user was able to establish the domain of the search; the search query supplied could be used to search a more specific search engine – metasearching. For example, specific queries for local products or services could be referred to directories such as Yell [YELL] in order to provide the user with localised results.

Furthermore, a user profile must be formed in order for each individual user to have web content mining performed on their behalf to provide recommendations. Several techniques are demonstrated by existing tools. Tools such as WebMate learn a user interest profile continuously and incrementally; as the profiling algorithm operates when a user explicitly marks a document as ‘I like it’. There are several pitfalls to this technique. Primarily, a user has to provide a certain amount of thought and decision making when marking a document. As this is done whilst browsing the web, interrupting a user’s trail of thought and concentration; this is a feature that is likely to irritate a user.

Lieberman [Lie97] supports this, suggesting that the simplest of interactions that last only a few seconds can be disruptive enough to a user that they will not bother to do it. Hence, the more a tool relies on user interaction to improve itself, the less likely it is to be used. There is also the possibility that as these recommendations rely on the user, human emotions may effect whether a user decides whether he or she likes the document, reducing the accuracy of recommendations.

Furthermore, if a user is only informing the system of positive documents of inclusion, the scope of suggestion becomes increasingly narrow and limited as parameters of the TF-IDF vector scheme need to be approximated. Schwab [Sch02] states that user interest should rely less on user ratings, rather taking a passive and unobtrusive approach to observing users in order to build its profile.

There are two general techniques used to model a user’s interest goal; a single global interest goal [Arm97] or multiple categories of interest [Lie97]. With the former, all interests recorded from the viewing of web documents are coalesced to form one single interest representation.
The main pitfall of this technique is that the assimilation of interests does not necessarily maintain a category of interest to the user. For example, if a user’s interests were ‘leather footballs’ and ‘cotton socks’, the union of interests ‘leather footballs cotton socks’ may produce documents regarding ‘leather socks’ as of interest to the user. This affirms that interests should be defined by multiple categories.

Due to the vast and varied information available on the Internet, there is a high possibility that a user’s interest may wander, perhaps due to an online advert or simply due to the human nature of discovery [Fal05]. However, this topic may only interest the user for a short space of time. Therefore, a user profile which indicates interests should incorporate a factor of decay [Lie95] due to the nature of a user’s interests and behaviour. However, this is not to say that this is not of interest to a user.

Most of the systems currently in use including WebWatcher, WebMate, Lira and Amalthaea have used the TF-IDF vector scheme to represent and filter documents. Furthermore HTML is used as a method of interface interaction. HTML has been used not only to perform keyword extraction, but also as a method of suggesting recommendations.

**Web Usage Mining:** In order to determine a user’s interest profile, many of the existing systems have relied on clustering. Currently, no personal web assistance tool has used contextual popularity to suggest to users other web documents of interest. Alexa is the only tool which promotes this technique, however this is implemented online, which would require a user interrupting his or her browsing in order to visit the Alexa web site and search for recommended links based on the URL of the document they were visiting. This technique would allow users to benefit from the experience of other web users. This is similar to Footprints [Wex99], but on a more global scale – as Footprints is site specific, only offering browsing paths within a web site.

The level of trust users have in a web site is also signified by the number of visitors a site has, as if there is a high level of usage for a website, it is highly unlikely the web site is a spam web site, or a web site with many pop ups. Equally, it is likely that the website is useful in providing the service it intends to provide.

Techniques for web usage mining are used in order to provide recommendations to a user. Footprints uses access patterns; however document recommendations can also be made by traversing links from a user’s current position as with Letizia, or by automating queries to a search engine as with PowerScout. However, the advice provided by the systems is not necessarily personal. Therefore, to create personal advice, the user’s profile can also be used to generate localised recommendations.

2.6.1.3 Recommendations from existing systems

Recommendations from the benefits of previous systems are:

1. **New document recommendations:** This technique is employed by Letizia, PowerScout, WebMate, WebWatcher, Lira and Amalthaea; recommending documents that are new to the user based upon the user’s interest profile. The new document recommendations are generated from traversing links from the user’s current position or by querying search engines using a combination of the user’s interest profile and the current document being viewed.

2. **Forward document path recommendations:** Demonstrated by tools such as Footprints and Alexa, the forward path browser make recommendations to users based on previous users’ access patterns.

3. **Popular document recommendations:** This tool recommends popular web documents to be stored as bookmarks; which are essentially shortcuts to web documents. This saves the user having to remember or type out the URL of the document itself. Letizia recommends documents to a user based on user interest similarities. Many tools extend this technique, using direct feedback from users to
assist the process of recommendation. Alexa and WebMate implement user feedback to assist recommendations.

2.6.1.4 Considerations
The problem web user’s face is that the results they expect when querying a search engine with a specific query is often not what is brought back. For this reason, contextual search has become an emerging technique.

1. Context: One tool, IntelliZap, has attempted to tackle the problem of contextual ambiguity. However, IntelliZap does not function as a search engine, only terms within a document already being viewed by a user can be searched upon. The user must highlight the term he or she would like to search for in the document and press a search button. The keyword selected, terms close to the selected keyword and META tags are used to define a domain for the document; which is used along with the keyword to generate contextually relevant results for the user to be recommended to view. As stated in 2.4.7, IntelliZap was proposed in a research paper, and the software itself is unavailable.

2. Location: Web users are also now beginning to use a plethora of smaller, geographic and domain specific local search engines rather than major commercial search engines to find the data they require [Men99]. This technique could be implemented within a software agent in order to find information such as services [YEL] and products [MIV] by querying more relevant sites.

3. Interest Estimation: As evident in 2.4, no current item of software which assists a user in their web browsing experience uses any method of establishing an estimate of a user’s interests prior to its use. The user profile is essentially built from scratch once the user begins using the software agent. A useful tool, although not essential, would be to estimate a user’s interests which can be built upon once user browsing has started. This would allow a software agent to perhaps weight documents which were part of a user’s interest estimation greater than other documents, as the agent is aware of a prior interest in such a topic. Several methods could be used to estimate a user’s interests such as parsing a user’s email inbox, web browser favourites/bookmarks, or personal documents.

There appears to be no client application in existence which assists a user in searching the Internet. Furthermore, with frustration lying with the fact that the results retrieved by search engines often being irrelevant, either by context or by location, a tool which assists a user to find exactly what they want is desired.

Furthermore, by aiding a user with recommendations and improving the relevance of the recommendations through making sure they are contextually and locally relevant would assist a users browsing experience and make their experience more efficient and enjoyable.

2.6.2 Implementation languages
A number of languages can be used to implement an agent for use with smart web browsing.

Current approaches to smart web browsing have used a variety of implementation languages. Letizia is implemented in Macintosh Common Lisp, WebMate in C and the other implementations using HTML as an interface for interaction with the client. By using HTML as a method of client interaction, it is obvious apparent that a further language must be used to implement the actual system.

Therefore by using an HTML interface, an extra overhead of communication between the HTML and system is required. By implementing the system within the same language as the interface, this communication overhead would be reduced.

However as the ability to integrate with HTML is required, it is essential that a language which is able to integrate with HTML is used. Furthermore, using a common language allows the
reuse of code, as a common code functionality can be shared amongst programs making error debugging easier.

The two platforms available for implementing such a system are Microsoft’s .NET architecture of Sun Microsystems’ Java.

<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 byte code, 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 source code twice during process of converting to native code. 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 7 – Comparison of .NET and Java architectures [INF]

The key advantage of .NET is that cross language development is encouraged. Therefore, a developer can use the language that suits the given module being coded; however still has perfect integration of the modules into a single application.

Additionally, .NET supports true code reusability whereas the Java framework does not. The .NET architecture also has a strong support for web services, whereas Java does support this feature yet not as comprehensively as .NET. Any software agent interacting with a search engine would require strong web service support.

The .NET GUI is also more familiar than the Java GUI to computer users due to the majority, currently 89.2% of computers, possessing Microsoft Windows as their primary operating system. Internet Explorer is also the web browser of choice for over 75.5% of Internet users for browsing the Internet [BRO].

Several other distinguishing features for the two architectures are shown in figure 8.

<table>
<thead>
<tr>
<th>.NET architecture</th>
<th>Java architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for web services</td>
<td>Good built-in support for web services.</td>
</tr>
<tr>
<td>Platform independence</td>
<td>Platform neutral code generated.</td>
</tr>
<tr>
<td>Code reusability</td>
<td>Advocates true code reusability, with good support</td>
</tr>
</tbody>
</table>

Figure 8 – Distinguishing features of .NET and Java frameworks [INF]

2.6.3 Usability

When designing user interfaces, it is essential that the HCI considerations associated with the design of systems targeted at web users are recognised. The user interfaces of such systems are critical to ensure that users are both able and inclined to use the systems. Hence Nielsens’s 10 Usability Heuristics [USE] must be referred to, to ensure the needs of a web user are met.

Hermans [Her96] reasons that users’ only use agents when they are able to support and aid them perform tasks, uninterested in their proactivity, adaptivity or benevolence. Therefore, the
final agent must suit the needs of a web user. In order to maintain HCI principles and to ensure that the final system does this, Nielsen’s 10 Usability Heuristics [USE] should be referred to.

1. **Visibility of system status** – the system should keep users informed of what is currently going on via feedback in reasonable time.

2. **Match between system and real world** – the system should communicate in a language familiar to the user (words, phrases, concepts), rather than in system speak of jargon.

3. **User control and freedom** – the system should allow users to leave unwanted states without having to go through extended dialogue, and undo/redo should be supported.

4. **Consistency and standards** – users should not need to wonder whether different words, situations or actions mean the same thing. Following of platform conventions.

5. **Error prevention** – rather than good error messages, a careful design which prevents errors from occurring should be used.

6. **Recognition, rather than recall** – objects, actions and options should be visible. The user should not need to remember information from one part of dialogue to another, with instructions for use of the system either visible or easily retrievable.

7. **Flexibility and efficiency of use** – accelerators, often seen by beginner users may often speed up the interaction for expert users such that the system can cater to experienced and inexperienced users. Users should be allowed to tailor frequent actions.

8. **Aesthetic and minimal design** – dialogues should not contain information which is irrelevant or seldom needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their respective visibility.

9. **Help users recognise, diagnose and recover from errors** – error messages should be expressed in plain language, with no code; should precisely indicate the problem and constructively suggest a solution.

10. **Help and documentation** – although a system is better if documentation is not needed to use it, it may be necessary to provide such help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to conduct and not be too large.

It is also useful to refer to Lieberman [Lie97] when addressing the concept of document recommendation. Lieberman suggests that recommendations from an agent are not critical decisions and should only act to benefit a user’s activities if they are worthwhile.

Therefore, suggesting recommendations rather than redirecting a user to a recommendation offers a partnership between the agent and user.

**2.7 Summary**

A detailed discussion on the subject of intelligent agents and the existing techniques used to implement such systems has been given. The current state of the art has been evaluated, with a comparison and critique of the system architectures, profiling methods and implementation languages used. The necessary HCI considerations which must be adopted in the implementation of an intelligent web system have also been discussed. These factors will all be taken into consideration when determining the specifications for an innovative approach to intelligent web browsing.
3 Specification

This chapter contains the specification for the implementation of a smart web browsing system. The specification formally states the requirements for the smart web browsing tool, and deliberates their implementation.

3.1 System overview

The objective of this project is to create a fully functional client-side intelligent web tool. The requirements of the intended resulting software can be broken down into essential and desired requirements.

3.1.1 Essential requirements

Essential requirements are functionalities which must be fulfilled by the final piece of software. The resulting software agent must meet the following requirements.

- Allows a user to search the web retrieving relevant results, taking into account both the context of the query, and the location of the user should that be necessary.
- Allows the acquisition of user data through the monitoring of a user's browsing habits and the learning of the user's interest profile completely autonomously without interruption to a user's web browsing experience.
- Models a user’s interest profile such that recommendations of new documents of relevance can be made available to the user as he or she browses the web in real-time. This is to be displayed to the user for selection and browsing.
- Allows bookmarks to be suggested to a user based on the frequency they visit particular web sites.
- Allow document shortcuts to be suggested to a user based on the frequency they follow particular access paths whilst browsing the web.
- Is a standalone application implemented to work on the Microsoft Windows operating system; either browser independent or able to be used in conjunction with the most popular web browser.
- A tool that makes use of user participation, however not obtrusively relying on the user’s web browsing experience.
- The final software product should adhere to Nielsen’s 10 Usability Heuristics.

Currently, the state of the art does not allow a user to search the web contextually. The proposed solution would be of benefit to all types of users, as search engine users have to modify their queries to retrieve relevant results over half of the time [Fal05]. Existing methods can be built upon to improve the current recommendation systems in place. Current research into web content, profile and usage mining, URL affordance and metasearching will be used to solve the contextual search problem and improve current recommendation systems.

As discussed in 2.6.1.1, a proxy server will not be used for the monitoring of a user's browsing activities. Therefore, IntelliWeb must itself monitor this through interfacing with the user's web browser. It is essential that the learning of user interests to construct and refine the user profile is completed autonomously, so not to disrupt a user's web browsing experience.

Recommendations, shortcuts and bookmarks must also be given to the user in real-time as human interests vary considerably over time. Recommendations or shortcuts at one point in time cannot be assumed to be of relevance to a user later on in their browsing unless
supported by their current interests. Recommendations, bookmarks and shortcuts should also be made in unobtrusive and evocative manners and not interrupt a user’s web browsing experience.

With the Internet such a recurrently used resource, any piece of software used in its conjunction must be unobtrusive, easy to learn and easy to use. Hence, IntelliWeb must also meet this criterion, with Nielsen’s 10 Usability Heuristics explicitly defining how to do so, to ensure users truly want to use IntelliWeb to enhance their web browsing experience.

Additionally, with Internet users becoming more aware of security problems and the risks associated with browser modifications [Dar06] along with the use of more prudent firewalls, it would be beneficial to create a standalone application rather than a modification to an existing web browser. This ensures users are both able and more inclined to use the final system.

Furthermore, IntelliWeb must either be browser independent or able to function in conjunction with Internet Explorer, which is currently the most popular web browser, holding 75.5% of the web browser market.

3.1.2 Desired requirements
The desired requirements of the final piece of software are functionalities which are not necessary, however are directly related to the essential requirements. It is desired that the resulting software agent meets the following requirements.

- The autonomous calculation of a user’s location, without the user needing to converse with the system.
- The user being able to modify his or her current interest profile should the recommendations not quite being what the user is expecting.
- User data monitoring should be easily deactivated.
- Unobtrusive messages to the user of recommendations, shortcuts and bookmarks; easily configurable by the user.
- The specification by the user of the number of results to return for contextual and localised searches, and for recommendations.

With the autonomous learning of a user’s location, localised results can be given without the need for user interference. However, the user should be able to modify this location in case the user’s computer is operating through the use of a proxy, or the user simply wants localised results for differing locations. Similarly, the user should also be able to adjust the user profile currently representing the user’s current interests in order to give more specific, targeted recommendations should they not be of satisfaction to the user.

The remaining desired criteria are all to do with making the system user friendly and configurable to the needs of the user. In line with the HCI requirements discussed above and to make the final system user friendly, the user should always be in control of the software agent. Hence, the user is able to deactivate data monitoring, enable and disable messages for bookmarks, recommendations and shortcuts, and is also able to specify the number of results returned by the IntelliWeb system.

3.1.3 Extensions
Time permitting, the following extensions could be added to the project. These functionalities are not directly required however will enhance the functionality of the final piece of resulting software.

- **Real-time Personalised News & Information**: Presently a user must specify the RSS feeds he/she wishes to view online to a news aggregator which then displays the news contained within that feed to the user. However, not all of the information
presented to the user may be of relevance. Therefore, through the use of a user’s profile and current interest profile (learned by the software agent in real-time), up-to-date news and information can be displayed to the user from the RSS feeds added to their profile.

- **Interest Estimation**: The estimation of a user’s interests before use of the final software tool. Each of the software agents currently in the public domain build a user’s interest profile from scratch once the user begins using the software agent. Although not prescriptive, the estimation of a user’s interest profile would allow a software agent to weight documents which overlap with a user’s interest estimation relatively greater than other documents, as the agent is aware of a prior interest in such a topic. Several methods can be used to estimate a user’s interests such as parsing a user’s email inbox, web browser bookmarks, or personal documents.
4 Decisions & Technical Challenges

Following on from the specification and to meet the requirements discussed, this chapter identifies and discusses the key decisions made in the scope of the project and the technical challenges involved in their implementation. Approaches examined in the background are evaluated against new techniques and approaches proposed.

Discussions in the background which can be found in section 2 of this report have examined the current state of the art. Further consideration was given to the beneficial features from these systems as well as further functionalities which would prove advantageous to a new intelligent web browsing system.

In addition, the background also highlighted that decisions involving the tools to employ within the system, web mining techniques to implement, system architecture and usability of the implementation need to be addressed.

4.1 System tools

As concluded in section 2.6, there does not currently exist a client application which assists a user in searching the Internet. Furthermore, with frustration lying with the fact that the results retrieved by search engines are often irrelevant to a user’s query, either by context or by location, a tool which assists a user to find exactly what they want is desired.

In addition, no current system offers a collection of personalised tools. Current tools provide either new web document recommendations, recommendations of web shortcuts or offer web document bookmarks. With the dawn of Web 2.0, by aiding a user with personalised recommendations, bookmarks, shortcuts and personalised news and information, the demand from both novice and experienced web users can be met.

Therefore, I propose the incorporation of the tools discussed below, into a single end user interface.

4.1.1 Metasearch: contextual and localised marginalisation

As discussed formerly, IntelliZap [Fin01] is the tool which comes closest to allowing a user to search the web contextually. IntelliZap allows a user to search, however is bounded by the terms present in the web document the user is viewing, using the document META tags to determine the documents context. Although published in a research paper for WWW10, the software agent itself is not available in the public domain and cannot be traced.

With the demand for personalisation and the advent of Web 2.0 as discussed in 2.5, there is a demand for search results to be both contextually and locally relevant. Users would find a tool which was able to give completely relevant results to search terms essential to their browsing activities. Users should be able to state the domain of a given search term; with the search query and domain then used to target more specific search engines – also known as metasearching.

Through the system providing a metasearch facility, users would no longer have to visit a plethora of search engines in order to receive localised and relevant results and would make web search simpler and more effective.

For example, results could be given to the user based on the country they reside in, to decrease the probability that the results received are irrelevant. Specific queries for local products or services could also be referred to local directories to provide the user with localised results.
4.1.2 New document recommendations
An array of tools as identified in the critique of existing web tools, provide users with recommendations of new documents. However, each of the tools proposes recommendations of new documents of interest to a user in varying manners.

Letizia [Lie01] uses a method known as forward prediction, based on a user’s current location. However this technique is only as effective as the bandwidth available to forward search from the user's current location. Therefore, this may take longer with low bandwidth connections. Furthermore, this technique generally has a limited amount of time before a user navigates to a new web page. Therefore, methods such as search engine querying as used with Amalthaea and PowerScout would prove more beneficial.

Amalthaea [Mou97] and PowerScout [Lie01] use the technique of querying search engines with automated queries determined from user interest profiles to provide recommendations of new documents of interest. This method is advantageous as it makes use of minimal bandwidth, hence is able to work with both high and low bandwidth connections. Also, search engine querying is faster that manual searching, as is done with Letizia. Hence, shorter delays will be received by the user before recommendations are made, a crucial aspect of any web based system.

4.1.2.1 Localised recommendations
The relevance of recommendations made can be improved through making sure recommendations are locally relevant. For example, if a user is looking for estate agents in Glasgow, it is of no use displaying to the user recommendations of estate agents in Sao Paulo. This improvement would assist a user’s browsing experience and make their experience more efficient and productive.

4.1.2.2 Temporal recommendations
The fact that a user’s interest profile will change over time must also be taken into consideration, and the fact that subjects that were of interest to the user at one point in time may not still be of interest to the user. These subjects should not remain the focus of a user’s interest profile and recommendation offerings, and should gradually be moved out of the user’s interest profile as new subjects are encountered and dominate the user’s browsing habits.

4.1.2.3 URL affordance
With an abundance of web sites cheating the index in order to receive greater traffic numbers, URL affordance will play a key part in ensuring that the reputable brands, services and domain names expected by the user are retrieved. Through the use of URL affordance, the quality of the web sites retrieved can be determined.

4.1.3 Personal shortcut recommendations
As identified in the critique of existing web tools, there currently exists a tool, Footprints [Wex99], which recommends shortcuts to web users to different web sites. These recommendations are based on the access patterns of other users. However, a tool which provides recommendations of shortcut to other web sites in real-time based on the current user’s web page access patterns has not yet been implemented. Through this tool, users will be assisted in browsing web sites they commonly visit in a more efficient manner. Such a tool is desired as it is rare for a large group of web users to completely share browsing access paths. Hence, through shortcuts to web pages being based on the user's browsing paths in real-time, shortcuts are more likely to be relevant to the user and effective.

4.1.4 Bookmark recommendations
Inexperienced web users learn through experience the benefits of bookmarking regularly visited and popular web sites. Experienced web users are perhaps also guilty of not using bookmarks to their full extent. This is represented by the fact that twenty percent of web users have difficulty in retrieving pages they have visited before [Bou01]. Therefore, all types of web
users would benefit from a tool which recommended their regularly visited pages as bookmarks.

4.1.5 Personalised news and information

Current aggregate newsreaders [RSS] currently present a user with all of the news articles listed under a given RSS feed. There is currently no personalisation or tailoring to a user’s interests apart from the subject area of the news feed. The subjects of RSS news feeds can be quite broad. Therefore, as a user increases the number of RSS news feeds they are subscribed to, the user is left to trawl through an increasing number of articles to discover articles of interest and relevance.

All types of web users would benefit from a tool that was able to determine articles of interest based on a user’s interest profile. This would save users the burden of having to scan across numerous articles, saving time and making a users browsing experience more effective.

4.1.6 Interest estimation for immediate personalisation

The current set of tools available including Letizia [Lie01], Amalthaea [Mou97] and PowerScout [Lie01], all learn a user’s interest profile from scratch. Therefore, initial recommendations based on a user’s browsing patterns are usually relatively broad and not wholly what the user requires. Thus a tool allowing the estimation of a user’s interest profile would allow a software agent to favour documents which overlap with a user’s interest estimation compared to other documents, as the agent is aware of a prior interest in such a topic. Furthermore, news and information displayed to the user can immediately be personalised to the prior interests of the user.

Several methods can be used to estimate a user’s interests such as parsing a user’s email inbox, web browser bookmarks, or personal documents. There are many privacy issues which are raised with the use of a user’s email inbox or personal documents, which were most effectively highlighted in the case of Google’s Gmail [GMA]. Therefore, as favourites/bookmarks are public and rarely contain sensitive information, any estimation of a user’s interests should use a user’s favourites/bookmarks.

This interest estimation would not be prescriptive, and would be built upon whilst the user continues browsing, in order to refine the user interest profile and offer more relevant recommendations and personalised news and information.

4.2 Techniques to implement

The set of tools determined formerly in 4.1 entail considerations of autonomy, document representation, user interest modelling and access pattern modelling.

4.2.1 Autonomous operation

Systems such as WebMate [Che98] ask users to specifically rate pages in a positive or negative manner. It is currently unproven that such a ratings based learning approach employed in such systems performs better or worse than autonomous machine learning. However, from a user perspective the process of assigning ratings can be arduous for the user to complete. Lieberman [Lie97] further addresses the concept of document recommendation supporting this, suggesting that the simplest of interactions that last only a few seconds can be disruptive enough to a user that they will not bother to do it.

Consequently, the more a tool relies on user interaction to improve itself, the less likely it is to be used. There is also the possibility that as these recommendations rely on the user, human emotions may effect whether a user decides whether he or she likes the document, reducing the accuracy of recommendations. The usability, performance and consistency aspects of an autonomous approach thus far outweigh those of a ratings based approach. Therefore, I propose implementing an autonomous method to learning a user’s interest profile opposed to the use of user ratings, thence minimising the burden on the user.
4.2.2 Document representation

Documents viewed by web users have been modelled effectively using the Vector Space Model [Sal83] by the majority of systems which comprise the state of the art. Similarity between documents is then calculated using the cosine similarity measure. Davison concludes that textual based similarity predictions through the use of a document vector far outperform mathematical approaches due to their simpler approach [Dav02]. In the use of Vector Space Models, documents are represented by (term, weight) vectors, where the weight is calculated by the TF-IDF formula as discussed in 2.3.1.

It has been demonstrated that the normalisation of the weight of each term within a document vector causes a severe improvement in document representation quality [Bal95, Kar00]. Using raw weightings to represent a term’s importance in a document can cause significant differences when calculating the similarity between a term and its most similar documents. This is because the weighting assigned to a term is based on its occurrence, and not as a percentage. Therefore, even though a term \( t \) could occur in two separate documents, \( x \) and \( y \), with the same percentage value, if the second document, \( y \), comprises of five times the number of terms than in document \( x \), the TF-IDF weight generated for document \( y \) would be a value five times bigger than that calculated for the same term in document \( x \). Through the normalisation of the weights of each term by the number of items in the document vector, a more accurate representation is given to the importance of each term within the vector.

With the use of a Vector Space Model to represent documents, where documents in this case are web pages; there is a loss of document hierarchy and therefore the document is not modelled completely accurately.

However, this does allow a range of techniques to be employed on the vector itself. Simply analysing the weighting of terms within a document to determine the meaning of a document would be insufficient, due to the utilisation of grammar; with the use of prefixes, suffixes, agentives etc, conjunctions and the nature of the English language [Por80]. In order to accurately determine document meaning, the removal of irrelevant words from a supplied stop word list [STO] can be conducted.

After this stage of processing, techniques such as the Porter Stemming algorithm [Por80] can be applied to the document vector to reduce terms to their meaningful stems. Although there is a marginal loss of document accuracy, this is insignificant compared to the performance enhancement gained through the storage and access to a smaller list of terms.

Furthermore, storage of the full document vector used to represent a document is not needed. Instead, the top \( N \) terms (where \( N \) is a variable to be decided); of a weight-ordered document vector can be used to represent the vector. After the \( N^{th} \) term in the document vector, the weights of the terms following are negligible compared to other terms in the vector and hence do not pose any advantages when computing the cosine similarity between itself and another document vector.

In addition, through the use of a cropped vector, the complexity of calculating document comparisons is reduced, due to fewer computational comparisons between vectors when performing similarity metrics.

4.2.3 User interest modelling

In order to provide personalised recommendations to a web user, it is essential a web user’s profile is modelled accurately. Rarely do web users have one category of interest. This is illustrated by the fact most web users have a number of web sites they visit regularly, most often encompassing a spectrum of subjects. Therefore, it is paramount to model a user’s interest profile using a number of categories of interest. A category can be defined as a set of documents where the similarity between each document vector and the category vector is above a given threshold value.

Documents of irrelevance to an interest category can further be defined as documents whose similarity differs greatly from the documents present in a given category. Hence if a document
viewed by a user does not conform to any of the current interest categories, the document can be termed as being irrelevant. The similarity between the irrelevant document and categories of interest will be below the set threshold. This threshold can be altered in order to adjust the flexibility of allowing documents into an interest category.

A user’s interest in a topic is rarely represented by a single theoretically optimal document. Hence the centroid of an interest category is often represented by the average of the document vectors contained within it. However, by using such an approach, previous interest categories that were once of interest to a user are discarded as the user’s new interest dominates his or her browsing activities. Furthermore, by having to store the document vectors for the documents already seen, the amount of memory consumed by such an approach quickly increases.

Advertisements and irrelevant information [Che98] in a document can lower the accuracy of a user’s interest representation. These two factors can also signal a change in a user’s web browsing behaviour. It is possible that the document viewed by the user is one that the user has randomly come across either by accident or intentionally; yet irrelevant to the user’s interest thus probably never accessed by the user again. This is known as a solitary access document, as the document itself or any similar documents are not viewed again. There is also the chance that such a document could identify a change in the user’s interests, as new interests do not necessarily have to overlap a user’s prior subject interests.

Two separate solutions to this problem have been utilised by user profiling algorithms. The first, more simplistic solution, disregards a document if it does not match a current category. Any document which does not match to any of the current categories is treated as a solitary access document. By doing this, new categories of interest are not formed as the document observed does not overlap any of the current interests held for the user. This is highly ineffective as new interests do not have to overlap previous interests and brand new unrelated interests will never be formed.

The second solution proposed in [Che98] always includes a new document into a category, averaging the two most similar documents if there is not enough space present in a cluster. However through this method, even irrelevant documents are included in a category and hence detrimentally effect the representation of the user profile. The benefit of this method is that documents which form the basis of new interests do not have to overlap previous interests in order to be recognised.

No existing solution is able to deal successfully with irrelevant documents and documents representing the formation of new interest categories. Therefore, the definition of a novel method is essential to handle irrelevant documents and documents representing the formation of new interest categories. Two new methods were investigated, with the latter chosen based on its merits over the former.

The first method investigated was the use of an N dimensional representation of the documents viewed by a user. Documents are represented by a top-N term cropped vector. As the user browses the web, a new dimension is added to the representation space to represent the viewing of a new document. The similarity between the new document and previously observed documents is then calculated for the terms featuring highly in the documents viewed.
so far. New categories of interest can be easily identified through the clustering of terms between documents, identifying a similarity between documents.

However, a major problem in the implementation of this method would be that as a user views an increasing number of documents, the number of dimensions in the space would increase. Therefore, an increasing amount of memory would be consumed as the number of dimensions increased; due to an increased number of similarities needing to be calculated. These calculations would be further increased by the increase of the number of terms stored in the top-N term cropped vectors. Furthermore, due to the modelling of the documents in multiple dimensions, the methodology becomes harder to envisage after the number of dimensions exceeds three.

Due to the complications of the former method, I investigated a second method based on the top-N cropped term vector scheme used to represent a document. In order to explain the incorporation of a new page in a user’s profile, it is worthwhile to reiterate that documents are represented as top-N cropped vectors. These vectors are of the form (term, weight); where the weight is calculated using the TF-IDF measure and is normalised.

The proposal makes use of a new devised scheme which can be likened most to a dynamic histogram. The persistence of terms is measured to represent a user’s current browsing habits. These terms are extracted from the documents being viewed by the user. When a document is viewed by the user, the top-N cropped term vector is used to determine its meaning.

If during a browsing session the documents being viewed are of the same interest subject, the terms used to depict these documents will remain roughly the same. Therefore, as the user’s interest profile is determined from the top-N cropped vectors for the documents viewed, the interest profile will vary only slightly. Due to the normalisation of the cropped TF-IDF vectors, it is relatively simple to determine the most representative terms for a document, and therefore these can be incorporated into the user’s interest profile.

As a user’s interest changes, the terms that will feature in the new documents being viewed will begin to change from those currently representing the user’s interest profile. Over time, as the user’s interests take a new direction, the subjects of the more recently viewed documents are incorporated and begin to dominate the user’s interest profile, better representing the user’s current interests.
The figure below depicts this pictorially, showing how as user’s interests take new paths, the terms representing the new documents being viewed begin to dominate the user’s interest profile. Each bar depicts a separate term.

The new proposal will also make use of a garbage collection facility to remove irrelevant documents and categories of interest which are no longer of interest to the user. As the user browses the web, the number of terms used to define the user’s profile increases rather than replacing the current terms used to represent the user’s interest profile.

This allows the resurgence of any old interests the user may have had, should they continue browsing in a former subject area. Should these interests be of irrelevance and no use to a user however, they will be eventually removed, giving a more accurate representation of the user’s profile. On the other hand, should these documents actually be representative of a change in the user’s interests, as the user begins to follow this interest, these interests will take precedence over other subjects. As this trend continues, old topics of interest can be removed safely.

By using such an approach, old topics of interest and documents of potential irrelevance can be put in a dormant state. Ergo, if a user is to view a document which is of the same category as an old topic or document of irrelevance, the terms used to represent such documents can begin to increase in weight and if these interests are followed, can eventually represent a user’s interests more succinctly. Documents which do not fit a current interest can be put in a dormant state, and removed from the profile completely after a given duration of time in which the interest was not followed. A dormant interest can be defined as the terms representing an interest which are below a given threshold.

An element of exponential decay will be applied to the terms representing the user’s current interest profile in order to gradually demote topics which are not of interest to the user. This method also allows the instant promotion of these topics should the user’s interest in these topics suddenly begin again through the observation of semantically similar documents.

This instigation of the garbage collection procedure can be based on the numeric weighting of the terms representing the irrelevant documents or interest. Due to the exponential decay applied to the terms within the interest profile, those interests not maintained will begin to decrease in weighting, eventually being removed when their weighting falls below a set threshold.

The diagram below depicts the modelling of new interests and garbage collection to remove irrelevant documents and subjects which are of no interest to the current user after the viewing of \( X \) documents, when the weight of several terms representing a user’s profile fall below a set threshold.
Due to the wandering nature of human attention, subjects at the forefront of a user's browsing activities at one point in time do not remain the focus of a user's interest profile if the user's browsing habits change direction. These subjects are gradually moved out of the user's interest profile as new subjects are encountered and dominate the user's browsing habits.

The proposed algorithm compromises of the following steps.

1. **Category establishment**: If the current document is the first document viewed by the user, the top-N terms of the document are used to represent the user's interest. Else, if there is already a new interest and the current document's subjects overlap with the current interest profile, the appropriate terms representing the interest profile and incremented. This measure is of range (0, 1).

2. **Demotion to dormant status**: If the current document is determined to be of a completely new subject to those representing the user's current interest profile, the document is demoted to dormant status and held in a buffer. Furthermore, if the terms representing an interest are

3. **New category establishment**: If the current document is determined to be of a completely new subject, the document is demoted to dormant status, and held in a buffer.

4. **Application of decay**: The terms representing a user's interest profile are subjected to an element of decay, (for example being multiplied by a decreasing fractional value). This ensures interest terms no longer of relevance to a user are gradually weeded out.

5. **New category promotion**: If the current document is determined to be of a subject that coincides with the subject matter of the document vectors held in the buffer, the document is promoted to active status. These terms are then represented in the user's interest profile accordingly.

6. **Garbage collection**: If any of the terms currently representing a user's interest profile fall below a set threshold, these terms are removed from the representation of the user's interest profile.

The new interest profile generated for the user can then be used to metasearch for new documents of interest and tailor up-to-date news to the user's current interest profile.
4.2.4 Access pattern modelling

In the determination of access path models, the consideration of the content of documents is not needed. Instead, the web sites addresses visited by a user and the ordering of these accesses are crucial to represent user access pattern models.

In [Che98], Chen’s maximal forward path algorithm provides a method of establishing the access patterns of web users. However, in Chen’s maximal forward path algorithm, only the unique browsing paths of a web browsing session are considered. In the context of web user access patterns, if duplicate access paths are included, a more accurate representation of repeat patterns can be determined.

In [Nan01], Nanopoulos suggests the use of graph traversals in determining access patterns, which refers to the available hyperlinks from a document to other documents. Current web users do not follow this methodology, visiting web sites irregardless of whether there are links between the web sites the user visits. Hence to model a user’s web access patterns more accurately graph traversal can be ignored. This models more accurately a truly interconnected web.

Furthermore, with many users possessing web browsing habits which demonstrate a large degree of noise, it is essential that sub-paths within a user’s browsing access patterns are also considered. In the context of the web, noise is the accessing of random web sites within a common browsing pattern.

The access patterns of a user will be stored as sessions for future retrieval. From the user’s browsing sessions, frequent sub-paths can be determined. Frequently accessed pages will be used to determine bookmarks, regardless of the pages accessed before or after these pages. The saving of these pages is of benefit to the user for quick access to popular web pages. Frequent subsets will be used to determine shortcuts. The benefit of such a tool is best explained in an example.

In the above example, the user has a frequent access pattern of \{WABXCYDEZ\} where A, B, C, D and E are pages that may differ on a regular basis. However, due to the nature of the web sites visited, the frequent access pattern is of the form \{WXYZ\}.

Therefore, when a user visits a document \W, it is not just of use to use the maximal forward access path to recommend document \Z. Documents \X and \Y can also be offered as shortcut recommendations to the user. This helps to make a web user’s browsing experience most effective. It should be noted that these shortcuts should only be offered if \{WXYZ\} is indeed a frequent browsing pattern the user follows. To determine this, the number of occurrences of the browsing access pattern must exceed a minimum support level.

Recommendations of shortcuts and bookmarks must also be given in real-time so that documents of relevance are offered to the user. Therefore, I propose the use of a new algorithm to determine shortcut and bookmark user recommendations.
The proposed algorithm compromises of the following steps.

1. **Repeat refresh abuse verification**: The current page is checked to ensure that the page being viewed is not the same as the previous page viewed. This is known as repeat refresh abuse. Some pages have automatic refresh counters embedded within them, and it is possible a user may be performing repeat refresh abuse unintentionally.

2. **Generation of browsing sessions**: A browsing session will be denoted as the viewing of $X$ number of documents. If the new page to be viewed will be the $X+1^{st}$ document, a new browsing session is created, and the previous session is stored. A limit must also be set on the number of sessions stored, also known as the log. This is due to storage and computation limitations, and also allows the removal of old browsing access patterns. Thence a user’s current access patterns are more accurately represented.

3. **Page appendage**: The page viewed is added to the new session if a new session was created, else is appended to the current browsing session. This continues until a session is completed.

4. **Calculate forward paths**: When the user has viewed enough unique pages such that the current session is now full, the forward paths of the sessions stored are calculated. If a session is not complete, only step 4c is conducted.
   
   a. **Generate sub-paths**: The sub-paths of each session in the log are calculated.
   
   b. **Aggregate repeated patterns**: Common sub-paths are aggregated in order to give an overview of frequent browsing patterns. Through this, frequented access paths can be stored.
      
      i. Sub-paths will be pruned to leave only those sub-paths above a minimum support level, according to equation 7 below. This allows noise to be filtered out and only frequent access patterns to be used to determine shortcuts.
   
   c. **Identify frequented documents**: Singly frequented documents are isolated in order to give recommendations of bookmarks to a user. If only this step is conducted, the number of accesses to this document is added to the number of accesses calculated from the previous forward path calculation.

5. **Set prior page**: The current page identifier is set to the current page, in order to detect repeat refresh abuse.

6. **User notification**: The user is notified of any bookmarks that may be of benefit. The user is also notified of any shortcut paths that may benefit the user when the user visits a document present in any of the stored sub-paths calculated previously.

\[ S_k = \{ \text{subpath} \mid \text{subpath} \in \text{BA}_u, \text{subpath.Count} \geq \text{min Support} \} \]

**Equation 7 – Pruning of sub-paths to meet minimum support**

$S_k$ is the final list of sub-paths, where the sub-path features in the user's browsing access patterns (BA$_u$) and the support exceeds the minimum support level.

**4.2.5 Metasearch**

The current top three search engines ranked by amount of traffic received each day are Google, Yahoo! and MSN [GOO, YHO, MSN]. Although the majority of web users frequent these three search engines, users also visit other smaller search engines for their web
These search engines are often queried by users for services and include search engines such as Miva [MIV].

Therefore, by implementing a metasearch function, users no longer have to visit a plethora of search engine web sites in order to find the results they need, saving time and increasing search effectiveness. By specifying a domain for a given query or a location (which will be determined autonomously from the web user’s IP address), users can be returned more relevant search results minimising the probability of spam web sites and web sites of irrelevance creeping into the returned results.

As no current metasearch or contextual search facility exists, and with web ranking algorithms kept highly secret, a heuristic algorithm needs to be developed in order to query the search engines which make up the metasearch engines; and to rank the results when they are retrieved so they can be displayed to the user.

Following on from my research I propose a new algorithm which makes use of current search engines and their results. However elements of context, metasearching, URL affordance and user profiling will be used to determine contextually relevant results for a user’s given query. The proposed algorithm compromises of the following steps.

1. **Input:** User supplies system input: search query $S$, and the domain for the search $D$. There is also an optional Boolean input which indicates whether the user wants local results. Should the user require local results, the user’s location $L$ is retrieved from the user’s profile.

2. **Conduct search:** Metasearch of relevant national MSN, Yahoo and Google search engines is conducted; with Miva & Yell also queried if local results are selected, with query $S$ to retrieve a list of results $R$, treated as an unordered list with any duplicates removed and noted.

3. **Vector calculation:** For each search result in $R$, the document retrieved is parsed in order to calculate two top-$N$ cropped TF-IDF vector of weighted terms, $V_d$ for the document content, and $V_h$ for the document TITLE, DESCRIPTION and KEYWORD META tags.

4. **Context filter through keyword density:** The cosine similarity metric is used to compare similarity between $D$ and $V_h$ for all results in $R$, or $L$ in $V_h$ if localised results were selected. Results sorted in ascending order by keyword density.

5. **Keyword density calculated:** Occurrences of $S$ in $V_d$ for all results in $R$ calculated. Results are sorted in ascending order by keyword density. Return results with keyword density between 4%-6% based on research into keyword density ranking.

6. **Generic re-ranking:** Cosine similarity used to calculate similarity between URL and union of $S$ and $D$ to re-rank the results in accordance with URL affordance research.
7. **Local filter:** If the user has chosen to receive local results, cosine similarity is used to compare $L$ and $V$ for all results in $R$, re-ranked by the similarity in ascending order.

The algorithm stated uses the individual properties of search engine algorithms to ensure documents relevant to the search term are retrieved. The top 10 search results retrieved from each search engine will generally have around 5% occurrences of the search term within them [SER].

However, due to contextual ambiguity for many terms, the results may not be expected by the user as the user will already have a semantic meaning of the search in mind. Therefore, with the user declaring the domain and the similarity between the domain and search results calculated, the possibility of irrelevant results is dramatically reduced.

![Figure 15 – Diagram representing the specification and relevance refinement of a web search when a domain is declared](image)

Furthermore, calculating the occurrences of the search term and documents again within a prescribed range allows further filtration of the results. The cosine similarity metric is implemented as this is used by the majority of tools to determine similarity between vectors. Metasearching is used on all queries across the three main search engines to ensure that no search engine bias occurs.

In addition, metasearching is used when a user selects local services or product results in order to provide city-localised results.

Current tools including Amalthaea [Mou97] and PowerScout [Lie01] use the technique of querying a single search engine with a given set of terms in order to determine new documents of recommendation for a user. Therefore with the use of metasearching, recommendations can be further refined by analysing the results retrieved by the three search engines. Each result can be analysed to determine its ranking amongst the set of results brought back by the various searches and its occurrence across the various search engine results.

Search engines can be interfaced with in two ways. Firstly, through the web sites which provide an interface for them to be queried, found at the search engine’s web address. Secondly, through a search engine’s application programming interface (API) that enables applications to use the facilities provided by the search engine. APIs are the best choice for developers who want to add search engine facilities to their applications; hence APIs for the range of search engines mentioned formerly will be used to retrieve search results. The fact that most search engine APIs provide extensive support for .NET is a contributing factor to the choice of the .NET framework as discussed in 4.3.6.
4.3 System architecture

4.3.1 Real-time operation
In the situation where users are browsing the web, assistance to a user must be personalised to the user’s interests and browsing patterns. The fact that a user’s interest profile will change over time must also be taken into consideration. Due to the wandering nature of human attention, subjects once at the forefront of a user’s browsing activities should not remain the focus of a user’s interest profile and recommendation offerings should the user’s browsing habits change direction. These subjects should be gradually moved out of the user’s interest profile as new subjects are encountered and dominate the user’s browsing habits. Therefore, to achieve relevant personalisation, real-time recommendations rather than recommendations based on previous interest profiles or web logs must be given to the user.

4.3.2 Information collection
There are two major methods of monitoring a user’s browsing activities; by the monitoring of the user’s browser or through a proxy server. Although proxy servers offer architectural flexibility, they are frowned upon by a large majority of web users who disagree with the surrendering of trust and personal information [Som99].

Furthermore, such architectures are difficult to impose on many networks, especially networks in places of employment; as such networks commonly operate a prudent firewall. This explicitly blocks all types of web proxies and services unless otherwise stated in the firewall configuration. Hence the use of a proxy in order to monitor a user’s web usage would be problematic. Furthermore, users would also need to understand how to configure the proxy for use, which could cause further problems for a potential user.

Another approach is that of monitoring a user’s browser application. The use of browser monitoring no longer requires modification of browser code and places less of a burden on web users as it requires no involvement from them in order to function correctly. However, due to the large number of web browsers in use, it is difficult to ensure that any tool will be fully compatible with all web browsers. Therefore to reach as many web users as possible, it is essential that IntelliWeb is able to interface with at least the most popular web browser.

4.3.3 Distributed architecture
Due to the distributed components which are essential in the running of IntelliWeb, the system must be able to capitalise on the benefits of a distributed architecture whilst protecting itself against the pitfalls which are also attached to such a structure. A distributed approach requires the implantation and handling of messages between components and increased error handling due to factors which include the following:

- **Increased mean time before failures:** Due to the heavy load placed on search engines by web users around the world and the fact that a plethora of search engines are being queried, the average mean time before failure is increased. In the context of this project, failures are deemed as error replies from queried search engines due to it having too many users or simply due to errors or failures on the side of the search engines themselves. This is referred to as a site ‘going down’. As these sorts of failures are usually temporary, the resubmitting of a search engine query is often enough to overcome the failure and retrieve results.

- **Social disagreements between client and server:** Should the way in which any of the search engines take the search queries as input and send back the search engine results as messages change, the IntelliWeb tool would also have to be changed. Therefore, by implementing a modular design, the amount of work needed to adjust to a change in policy by any of the search engines can easily be handled.

Given the detriments of such a framework, there are also a number of benefits which include:
- **Increased flexibility**: Information regarding the users of the IntelliWeb system will be stored in remote location. Therefore, due to this, the user is flexible to use the tool on any computer as long as it has Internet access. Therefore the application is not restricted to working on a single computer.

- **Concurrent user access**: A distributed framework allows multiple users to access the storage facility of the IntelliWeb tool simultaneously. This is of benefit to both client and server as resources can be shared and the mining of data only needs to occur at one place.

### 4.3.4 Data storage

IntelliWeb must monitor a user’s browsing activities and store information such as bookmarks, shortcuts and RSS news feeds. It is evident that some form of data storage is required, either at the client or at the server side and will form an integral part of the system. In order to minimise the burden on the client, data storage on the server side would prove most beneficial. This would take up less disk space on the user’s computer and is less likely to be accidentally removed, modified or damaged by the user. Furthermore, server side data storage allows the aggregation and calculation of statistics such as access patterns and browsing paths which can be used to further benefit a web user’s browsing experience.

There were several choices for the method by which to store this information; through the use of native XML databases or a relational database. Native XML databases store data in structured XML files. XML files offer a tree like structure and are often easy to understand. Searching however, poses a problem. Creating, maintaining and using index documents can lead to a hefty overhead.

Complete XML documents may be returned by a search even though only a few attributes within each are required. This would lead to a greater amount of data being transferred between the server and user’s client machine. Furthermore, extra processing is required to access the required data from the retrieved XML documents. For the large amount of data which will need to be stored on each of IntelliWeb’s users, this would not prove as efficient as a relational database.

Relational databases such as Microsoft SQL Server [MSQ] offer a number of advantages over native XML databases. Searches can be conducted within relational databases much faster than can be completed in large native XML databases. behaviours can also be implemented in a relational database through stored procedures that can be invoked internally within the database and external applications. Relational databases also offer concurrency and transaction control, along with the enforcement of referential integrity.

Furthermore, there is vast support for relational databases and integration is simple and highly documented for the .NET architecture. Relational databases are also very stable with easy porting of code to other databases that adhere to SQL standards, and offer cross platform ability, with the ability to run on all versions of UNIX and Microsoft Windows [DAT].

### 4.3.5 Metasearch architecture

Communication between components of an intelligent software agent and various search engines would prove most successful if communicating using SOAP on a TCP transport layer. The reason for this is that most firewalls prevent the use of all ports except 80, the default port for HTTP. System administrators fear opening other ports could pose a potential security problem. Hence the use of SOAP under port 80 ensures that messages will pass through the firewall safely.

### 4.3.6 Implementation language

In alignment with the project context, Microsoft’s .NET architecture offers several crucial advantages over the Sun Microsystems’ Java framework; with each its own merit in the implementation of such a system.
- **Familiar graphical user interface:** With the Microsoft Windows operating system almost monopolising the operating system market, computer users are accustomed and familiar with its user interface. The .NET framework provides graphical user interface components akin in look and feel to existing Microsoft Windows products. This familiar environment is not offered by the Java framework, requiring greater adaptation by the user to the environment displayed to them. Therefore, through the use of the .NET framework, there is an increase in usability for the user.

- **Faster program execution:** In recent benchmark tests, the .NET framework was reported as being faster than the Java framework [BEN] although consuming slightly more memory that the Java framework when performing identical tasks. As discussed in 4.3.1, real-time usage is essential. Therefore, any increase in speed is beneficial when reporting real-time recommendations to a user.

- **Multiple cross-language support:** There are currently 30 different programming languages supported by the .NET framework. Therefore, with each programming language having its own benefits, different system components can be developed in the most suitable programming language, with communication amongst components remaining transparent. Furthermore, future expansion possibilities as discussed in 4.3.7 are enhanced.

- **True cross-language code reusability:** With the benefit of cross-language code reusability, the possibility of repeated code is dramatically decreased.

- **Excellent web service support:** The .NET framework offers several packages such as the Web Services Enhancements for Microsoft .NET (WSE) [WSE], a supported add-on to the Microsoft .NET Framework. Such packages provide excellent support for evolving web services protocol specifications. Furthermore, there is extensive documentation to help in the development of applications encompassing web services.

4.3.7 Future expansion

Due to the rapid evolution of the web, it is important to ensure that the system implemented is done so through the use of a modular design strategy, allowing the option for future expansion. The use of a modular design strategy allows the opportunity to add further components to the system without the need to make extensive changes to the current architecture.

4.4 System usability

System usability is essential for [USA] any application and is a necessary condition for survival. With so many different applications offering overlapping functionalities, leaving is the first line of defence for when users encounter a difficulty. Therefore, the final system should be as user-friendly as possible, especially with an application that intends to be used often, due to the recurrent use of the web.

4.4.1 User notification strategy

Lieberman [Lie97] states that ignorable recommendations and notifications to a user offer systems a higher degree of usability than the approach that acts explicitly on browsing suggestions. A co-operative relationship with the user must be maintained, without forcing recommendations upon a user, in order to build up their trust in the system. Therefore a recommendation strategy which offers recommendations as unobtrusive advice should be implemented.

4.4.2 Usability heuristics

As discussed in 2.6.3, Nielsen’s Ten Usability Heuristics [USE] offer guidance for conveying a highly usable interface to the end user, as well as tangible approaches for evaluating the final system in terms of its usability. Through the adoption of these heuristics, the final system can
be designed to adequately support the needs of the user and evaluated as to how well it accomplishes this.

4.5 Summary of technical decisions
Given the specifications in section 3, the technical challenges have been determined and the necessary theoretical decisions made to overcome these challenges. These decisions include new proposals for modelling web browsing habits and the novel idea of encompassing metasearch into one user tool.

Given that these decisions have been made, the design of the system can promptly begin.
This chapter details how the decisions made in regards to the technical challenges of the project were translated into the system design. A brief overview of IntelliWeb is given with each of the IntelliWeb components further reviewed in greater detail.

The final system has been named IntelliWeb. IntelliWeb intends to incorporate a plethora of intelligent web browsing tools which can be used by multiple concurrent users. The final system must meet the specifications as set in section 3, hence the design must incorporate these criteria whilst taking into consideration the decisions made regarding the technical challenges that are faced in meeting these specifications, as discussed in section 4.

5.1 System overview
The diagram below depicts the conceptual design of the IntelliWeb system. The directed arrows illustrate process communication which is to be implemented.

The diagram numbering depicts the dataflow through the system. Explanations of these interactions are given below, followed by the explanation of key design components of the final system.

1. HTTP request and response: When the user chooses a web address to visit, the web browser forwards an HTTP request to the WWW and receives a resulting response. The web page at the URL is downloaded to the user’s machine and displayed to the user.

2. HTML content logging: The current URL the user is viewing is parsed by the processing engine. This feature allows the monitoring of the users web browsing
habits, without compromising the user's trust in the system or disturbing the user's web browsing experience. The document is converted into a TF-IDF weighted vector, discussed in more detail in 5.2.

3. **Read and update user profile**: The top-N terms from the document vector are extracted and incorporated into the user profile. The user profile is refined to represent the user's current interests. This information is then used by the recommendation engine and news retriever which tailor recommendations of relevant documents and news articles to the user.

4. **Search engine**: Should the user choose to search the web, the user must specify a search query. For further contextualisation of results, the user can choose to specify a query, or a local search. The user's location is determined by determining the details of the user's IP through the WWW. When a search is made, Yahoo!, MSN and Google search engines are queried and the results retrieved re-ranked by a ranking heuristic before being displayed to the user. The search process is discussed in more detail n 5.3.

5. **Recommendations engine**: Changes in the user profile are reflected by the offering of new recommendations of relevant documents to the user. New recommendations can be determined by searching the WWW. This is discussed in more depth in 5.4.

6. **News article personalisation**: With the refinement of the user profile, articles of interest can be determined by and displayed to the user. The news articles to choose from are determined through the user's addition of RSS news feeds to his or her profile.

7. **Shortcuts tool**: Through the retrieval and analysis of the user profile, bookmarks are suggested for frequented websites. Shortcuts are also offered for web sites that are part of a user's browsing habits when following certain browsing paths.

8. **Recommendation following**: If a user chooses to follow a document suggestion, bookmark, shortcut, or view a news article, it is opened in the current browser window. If no browser application is active, the URL will open in a new browser window.

9. **Bookmark collection**: A user can export the bookmarks saved by the browser for importing into the IntelliWeb application. This interest estimation allows the refinement of the user's interest profile for immediate personalisation of recommendations and news articles.

### 5.2 HTML processing engine

The diagram below gives an understanding of the various processes involved in the processing of a HTML document. In a system such as this, where HTML is recurrently parsed, analysed and normalised, it was essential to create a dedicated engine that would handle these functions in order to ensure a modular and late binding design.

There are several HTML parsers available [HTM] which all create HTML tree representations of the current URL document being viewed. However, these parsers carry little documentation and often fail in the case of blank META tags. In addition, as I proposed in 4.1.2.3, the URL of a document must also be taken into account when analysing its relevance.

Therefore, I will code a separate HTML parser, which will analyse a document's HTML tags, HTML content and URL. The benefit of such an approach is that with an increase of the number of sites not setting META tags, the content and URL of a web document can be used to reasonably deduce the document’s meaning.

All HTML tags and other fragments of HTML and scripting languages are removed from the document to leave the pure core text of the document. The document is then compared to a
stop word list [STO] to remove meaningless words that would only serve as a detriment to interest profiling. The resultant representation is then subjected to the Porter Stemming algorithm [Por80] which reduces terms in the document to their meaningful stems.

The HTML document is checked for keyword and description META tags. If these tags are present, they are used to calculate a separate weighted TF-IDF vector for the head of the HTML document. The HTML body content is also used to calculate a weighted TF-IDF vector.

The benefit of two separate TF-IDF vectors is that the vector representing the HTML content can be used to refine the vector representing the head of the document, in order to accurately determine the semantics of the document. The amalgamation of these vectors is also an indication of the true meaning of the document.

![Figure 17 – Processing of a web document by the processing engine](image)

Weighted TF-IDF vectors serve to describe web documents, as discussed in 4.2.3. Only the top-N terms of the document need to be stored, both to reduce the amount of memory consumed and also as after a certain number of terms, the terms themselves are negligible in the depiction of a document’s meaning.

The cropped top-N document vectors can then be used to refine the user’s profile, allowing the recommendation of new documents of interest; and the refining of the news articles on display to the user.

### 5.2.1 Web browser monitoring

In order for the HTML contents to be extracted from visited documents, the web browser itself must be monitored. Therefore, the system must run alongside the user’s web browser. When accessing web pages, the browser sends an HTTP request and receives a response from the WWW. Once the request response has been received, the system can extract the HTML contents of the document currently being viewed by the user. This will then be analysed as discussed formerly. This process is repeated as the user accesses further web pages.
In order for the browser to be monitored through the application, the browser initialisation must be handled by the system. When a user inputs a web address to visit, or chooses to follow a search result or recommendation, this URL is automatically opened in the currently monitored web browser window. If there is not a web browser window currently active, a new browser window will be initialised by the system.

5.3 Metasearch engine

As the IntelliWeb system is to provide a metasearch tool to query the web, which can also be used to determine documents of interest to a user, the metasearch engine was developed as a separate module to conform to the need to allow future expansion.

The search engines will be queried using their respective APIs [MAP, GAP, YAP]. The APIs provided allow up to 1000 queries to their respective search engines a day. The use of an API to query these search engines removes the need to query a search engine through HTTP, where the results obtained would have to be parsed from the returned HTML. Due to the nature of web services as discussed in section 4, efficient error handling is necessary.

![Figure 18 – Metasearch engine architecture](image)

The user specifies a search query to the system, which is passed to the metasearch engine. The user can choose to specify a query domain also, however it is not necessary. The submittal of a query domain allows the refinement of the returned results to the user’s intended context, removing the likelihood of irrelevant results.

The location of the user is determined from the IP address the request was sent from. The user’s location can be edited manually also. However through this methodology the user is not interrupted in their web browsing, complying with the usability specifications discussed in section 4.4.

The query is then sent to the Google, MSN and Yahoo! search engines and the results from the respective search engines are received. The HTML head and body of each document comprising the returned results are retrieved, and passed to the ranking component.

The ranking component analyses the returned results for repeat occurrences of web addresses. Repeat occurrences are removed and flagged. Each result’s HTML head and body is passed to the HTML processing engine in order to calculate the weighted TF-IDF vectors. These vectors are then analysed using the query, domain and user location in order to determine a numeric weighting for each result.

Weightings are calculated using the keyword frequency of the search term, domain frequency, location frequency, associated terms, document URL, number of external links,
number of incoming links and the number of occurrences of the result across the three search engines. The results are then ranked on the basis of these weightings and then displayed to the user.

5.4 Recommendations engine

Recommendations are given to a user on the basis on their interest profile. The recommendations engine comprises of the suggestion function, and the offering of bookmarks and shortcuts to a user. It was designed as a separate module to comply with the necessity to allow future expansion.

The recommendations engine receives a list of the terms currently featuring in the user’s interest profile. To determine the terms used to query the metasearch engine, the difference in the weightings between the top pair of terms is calculated. If this is above a certain threshold the latter term is also added to the query. This is done until the difference in the pair of terms next calculated falls below the given threshold. The metasearch engine is then queried with these terms using broad search to offer a broader targeting option.

The top terms calculated by the recommendation engine are also used to refine the news articles displayed to the user.

5.5 User interface

The design allows the user to maintain absolute control of all aspects of the application. A user action (the clicking of a button) is enough to change the visible web page when a new document recommendation, shortcut or bookmark is offered. Furthermore, only two inputs are required if a user requires contextually relevant or local results. If too many clicks are needed to achieve a certain task [TRO], a user is less inclined to do it; therefore it is essential that the user interface is not over complicated.

The user interface will communicate with all system components. The user interface acts as the primary feature for all user input and output. All output will be displayed in the same window that the user inputs information to.
The user interface will be designed to meet the criteria of Nielsen’s 10 Usability heuristics. To evaluate whether the final system indeed meets Nielsen’s 10 Usability heuristics, the testing of the user interface will be conducted in the testing phase of the project.

The design will be implemented in a similar fashion to the diagram below. The numbers designated to various aspects of the design refer to certain usability heuristics as specified by Nielsen. An explanation of the heuristic referred to and how it is met is also given.

Not all of the usability heuristics are included in the diagram as some heuristics concern error messages, error prevention, help and documentation.

- **1: Visibility of system status** – the system keeps users informed of what is currently going on via feedback in reasonable time.
- **2: Match between system and real world** – the system communicates with the user in words, phrases and concepts familiar to the user rather than in system speak of jargon.
- **3: User control and freedom** – the system allows users to leave unwanted states without having to go through extended dialogue, and undo/redo should be supported.
- **4: Consistency and standards** – consistent, standard terms are used for system actions with the following of platform conventions.
- **6: Recognition, rather than recall** – objects, actions and options are visible. The user is able to use recognition rather than recall to navigate through the system. Instructions for use of the system are visible and easily retrievable.
- **7: Flexibility and efficiency of use** – menus act as accelerators, speeding up the interaction for expert users.
- **8: Aesthetic and minimal design** – dialogues do not contain information which is irrelevant or seldom needed.

**Figure 20 – Designed graphical user interface for the IntelliWeb application**
Authentication will be required in the form of a username and password. This both ensures privacy and allows concurrent users to be referenced uniquely. This can be in the form of a minimalist login screen when the user first launches the application. All functionalities of the program will be unavailable to the user until the user is authenticated. Should the user not have an IntelliWeb account, they must be able to make one. This can be done in the form of another screen, simply requesting username and password details.

5.6 System database
As discussed in 4.3.4, the use of a relational database was chosen over an XML database. The database will be stored on a remote server, to allow concurrent access. The use of basic login details allows interest profiles to be determined for users to offer recommendations of relevant documents, bookmarks, shortcuts and news article personalisation.

5.7 Interest estimation
The functionality of estimating a user’s interest was specified in the specifications for the IntelliWeb tool. This allows the recommendation engine to favour documents which may overlap with a user’s interest. News and information displayed to the user can also be immediately personalised to the prior interests of the user. This can then be built upon, rather than working from scratch in building the user’s interest profile.

Several methods can be used to estimate a user’s interests such as parsing a user’s email inbox, web browser bookmarks, or personal documents. A user’s bookmarks are public and rarely contain sensitive information; therefore, an analysis of the user’s bookmarks will be used to estimate a user’s interest.

As all browsers have different ways of storing bookmarks, a separate tool has been designed to allow a user to convert the bookmarks they have stored in their web browser into XML; which can then be read by the IntelliWeb client. Only the most recent bookmarks will be used to determine a more recent profile of the user.

The diagram below shows the functionality of this separate tool in conjunction with the proposed IntelliWeb design.

![Diagram](image)

**Figure 21 – Importing bookmarks to estimate user interests**

The user exports his or her bookmarks from their web browser of choice. This is then used as input to the bookmark collection tool in order to convert these bookmarks into XML, so that they can be parsed easily by the IntelliWeb system.
A user can then choose to import this XML document into IntelliWeb. By importing this XML file, the bookmarks are parsed, and the bookmark titles, date of saving and URL contents are analysed in order to establish the user’s interests.
6 Implementation

This chapter details how crucial aspects of the system were implemented and describes the problems which had to be overcome in their implementation.

All system components were developed in the Microsoft C# or VB .NET programming language under the .NET framework. A discussion of the implementation of key components is given below.

6.1 HTML processing engine

As examined in 4.3.2; due to the disadvantages of the use of a proxy server to monitor a user’s browsing habits, user monitoring was conducted through the manipulation of the user’s web browser. The diagram bellow shows a UML representation of the HTML processing engine.

Figure 22 – Unified Modelling Language diagram of the HTML processing engine

In order to create a cohesive and modular system, the HTML processing engine was created with the use of dedicated classes for the various functions of the engine. Due to the nature of the engine; in particular, methods such as the Porter Stemming algorithm and HTML parsing, the comparison of strings and string endings is necessary. Therefore, VB .NET was used to code these classes as the VB.NET string comparison function is faster than C# comparisons [MSD, MBR]. Furthermore, VB .NET offers support for late binding and simpler event handling, where a method can declare that it handles an event, rather than the handler having to be set up in code.

The user’s browser is monitored through the creation of a forked thread, which allows the HTML content of the document being viewed to be extracted. This content is then analysed to determine whether the page viewed contains META tags and identifies the head and body of the document.

The document is then passed to the HTML parser which strips the HTML document of all unnecessary elements, applies the Porter Stemming algorithm to terms over three characters in length, aggregates the terms and determines the normalised TF-IDF document vectors.
Through testing of the engine, it was decided that the top five terms of a document are sufficient to accurately model a document’s semantics.

6.1.1 Content extraction
The HTML parser created first identifies the head and body of the HTML document. The head is then examined for the keywords, description and title keywords in order to determine their contents. The HTML page is examined for the use of headings, and the contents of these headings are noted.

![Image](image.png)

Figure 23 – Example of head and body separation and the use of heading styles

Heading styles are used to denote important segments of pages [HTH]. The contents of these headings are used to determine whether the keyword, title and description tags of the document are similar to the content being discussed. Regular expressions are then used to strip the unnecessary HTML contents from a document. Terms with fewer than three characters are removed from the document and the remaining terms are stemmed to determine their true meanings.

The URL of the current document is also examined in order to determine whether it supports the semantics of the document. The terms are then aggregated and normalised to generate a weighted TF-IDF vector to represent the document.

The use of regular expressions to strip HTML contents is slightly slower than products such as [MIL]. However due to the deficiencies of [MIL] as discussed in 5.4, the advantages of implementing a new HTML parser far outweigh any minor delay in parsing. Furthermore, as HTML content parsing is conducted in a separate thread (discussed in greater detail in 6.1.3) there is no interruption to a user’s browsing.

6.1.2 Content analysis
The Porter Stemming algorithm used, which prunes terms to their meaningful stems, is available from [POR]. However, as discussed in the algorithm’s documentation, there are several shortcomings to the available algorithm. For example, words ending with ‘es’, such as ‘caresses’ can typically be trimmed to determine the meaningful stem by removing the ‘es’ if the third to last letter is a consonant.

However, with terms such as ‘mortgages’ this approach does not give the correct stem, leaving ‘mortgag’ as the determined stem, rather than ‘mortgage’.

To overcome this, the use of a separate array of terms and their meaningful stems was introduced for around thirty terms which did not meet with typical methodologies applied by the Porter Stemming algorithm.

6.1.3 Use of threading to prevent browser interruption
Upon implementation of the HTML processing engine there was a problem with the parsing of the HTML document being viewed by the user. The user’s browser window would become unresponsive for 2 to 3 seconds whilst the HTML content of the page was parsed. This was due to the use of a single thread for the system. With a single thread, method calls are
synchronous, and hence with large HTML documents, all other system functionalities await the return of the HTML processing engine before executing.

This delay was unacceptable, and did not meet the usability requirements specified previously in section 3. Therefore, to overcome this, a new thread was launched with the purpose of executing the HTML processing engine; so that the HTML processing engine could run asynchronously, ensuring no interruption was made to the user’s web browsing.

6.2 Metasearch engine

The search engines used to provide the metasearch facility were Google, MSN and Yahoo! [GOO, MSN, YHO]. Due to the nature of web services as discussed in section 4.3.3, efficient error handling was put into place to inform the user should any of the search engines fail; due to maximum server allowance (i.e. the search engine is currently conducted the maximum number of queries possible) or suffer any other failure.

6.2.1 User location

The location of the user was obtained by sending a ping request to a designated HTML page set on a remote server which determined the user’s location from the IP address used to access the web. Many tools exist to determine the location of an IP address, such as [IPL]. This was used to establish the relevant national search engine to query, for more relevant and personalised results.

6.2.2 Live access

A further functionality was added to improve the quality of the search results returned to the user. When the search results are received by the metasearch engine for a given query, each web address returned is sent a ping request to ensure that the web address is live and error-free. If the ping request fails, the web address is deemed inaccessible, and thus unable to be viewed by the user. This removes the possibility of the user navigating to a dead link.

Even though these failures are temporary, this often happens in search engines when web sites go down for maintenance or other reasons. If the ping request is successful, the web address is deemed live and the HTML head and body of the document is retrieved, and passed to the ranking component.

6.2.3 Quick search

As discussed in 5.3, the search results returned by the search engines are analysed by the metasearch engine to re-rank the results. The re-ranking is based on the keyword frequency of the search term, domain frequency, location frequency, associated terms, document URL, number of external links, number of incoming links and the number of occurrences of the result across the three search engines. The re-ranking procedure is applied to the both the head and body of each HTML document to ensure that the content of a document is of the same context that the head (keyword and description META tags and document title) imply.

As the body of a document can be substantial in size, documents with sizable textual bodies take longer to parse than those with smaller textual bodies. Therefore, a quick search procedure was implemented. Through the use of a quick search, only the head of an HTML documents is parsed and used to re-rank the search results.

Through the use of quick search, search results are analysed and displayed more rapidly to the user than the full, standard search. However, with the use of quick search, as only the head of a document is analysed, the possibility of spam web sites being displayed to the user are increased, as the body of the document is disregarded. Therefore, the user has the choice of a quick or standard search.

6.3 Recommendations engine

Recommendations to the user are made in the forms of:
Recommendations of new documents of relevance

Bookmarks

Shortcuts

News and information article personalisation

Recommendations of new pages based on a user’s interest profile are found by querying Google, MSN and Yahoo! [GOO, MSN, YHO]. This is done as the user browses the web, with the metasearch engine passed the top terms of the user’s interest profile to query with, as discussed in 5.4. If a user is dissatisfied with the types of recommendations made, they are able to view their current interest profile and signify terms they wish to be weighted higher, or disregarded from their profile. This allows greater personalisation and maintains user control at all times.

As a user’s interest profile is learned over a period of time, it takes time before accurate recommendations of new documents are made. Therefore, to ensure recommendations are as relevant as possible from the outset, a user can choose to import their bookmarks into the IntelliWeb tool in order to refine their interest profile. This was developed as an extension to the IntelliWeb tool, and is discussed in greater detail in section 6.3.4.

6.3.1 Bookmarks and shortcuts

Bookmarks and shortcuts were given a minimum support level of 3. Using this support level, a bookmark is only offered to a user if the number of accesses to a site is greater than or equal to 3. Shortcuts are offered in a similar way, if the access pattern occurs at least thrice in a user’s log.

It was proposed in the design section of this report that bookmarks and shortcuts be determined on completion of a user session. It may be useful to reiterate that a browsing session, in this context, is the viewing of $X$ number of documents. Upon viewing of $X$ documents, the user’s access profile is saved to the system database. Bookmarks are then calculated and offered to the user, and shortcuts offered when the user visits a web site in a particular browsing path. However, as recommendations must be given in real-time, this was modified such that a copy of the current bookmarks and shortcuts was also held in memory, so that bookmarks and shortcuts could be given during the course of a browsing session.

This is best illustrated in an example. Say it has been determined that a user has visited site X twice and frequented the access pattern XYZ twice during the course of their browsing. If the user had just started a new browsing session, and visited site X again whilst browsing, he or she would not be offered a bookmark to site X until the session had been completed, saved to the database, and the session data analysed. Therefore, by holding a copy of the current bookmarks and shortcuts frequencies in memory, this value can be automatically adjusted to represent the up-to-date browsing patterns of the user and offer real-time recommendations.

6.3.2 Suggestion alerts

As well as displaying recommendations of new documents, bookmarks and shortcuts within the user interface, it was decided that unobtrusive suggestion alerts could be used. This is discussed in further detail in 6.5.

6.3.3 News and information personalisation

The top N terms of the user’s profile are used to refine the news articles and information displayed to the user. The article title and contents are parsed for instances of the terms which contribute to the user’s interest profile. If any of the terms in the user’s interest profile are found in the article, the article is deemed relevant and displayed to the user.

The RSS feeds examined are determined by the user’s saved RSS bookmarks. Initially, when the user creates an account to use the IntelliWeb system, only the BBC world news [BWN] RSS feed is incorporated. The user is able to personalise the number of articles displayed, and the colours which denote the various news sources.
For increased usability, a functionality allowing the user to edit the articles displayed to them was added. Users are given the choice of having articles personalised to their interest profiles, or simply viewing all of the articles all of the time. This is useful in cases where the user’s interests become so abstract that personalisation of content leads to little or no articles being displayed.

### 6.3.4 Interest estimation

A user's interests are estimated through the use of the bookmark collection tool. As discussed formerly, this tool allows a user to define an interest profile so that recommendations can be made on immediate use of the IntelliWeb system, rather than after the duration of time taken to learn the user's interests. This interest profile is then refined as the user browses the web.

The bookmarks exported from the user’s web browser are converted into an XML file. XML [XML](https://en.wikipedia.org/wiki/XML) was used to store the bookmarks as parsing is simple, the file can easily be understood, and the file can easily be integrated with other applications if further work was to incorporate them. When imported into IntelliWeb, this XML file is parsed in order to create a TF-IDF normalised vector of the most recent title and description tags used to describe the bookmarks. The top-N terms from this vector are used to refine the user’s interest profile.

This tool also served the secondary purpose of ensuring recommendations for bookmarks which were already saved in the user’s web browser were not offered as bookmarks by the IntelliWeb tool. After importing the XML file created by the bookmark collection tool, the bookmark URLs are stored in the database and referenced before offering the user a bookmark recommendation. If the bookmark is already present in the database, the user is not offered the recommendation.

### 6.4 User profile and access patterns

User profiles and browsing access patterns are stored in the database. However, due to the recurrent access required to this information, it became evident that the frequent querying of the database resulted in inadequate system performance. Therefore, to solve this problem, caching was used. A cache of the user’s profile and previous sessions were held in memory for immediate access.

On successful launch of the application, the user profile and session log is retrieved. On exit of the application, the user’s interest profile and new session log is saved to the database. Furthermore, during use of the IntelliWeb tool, after certain durations of time the current user profile is auto-saved. This is a safety procedure put into place to ensure that even if the user’s application or computer were to crash, a recent version of the user profile would still be available.

To ensure that user details are saved correctly, a lock is obtained on the database whilst the profile is saved; to ensure no other access is permitted during this time which may compromise the data.

In addition, in order to allow the user to maintain control at all times, the added functionality of disabling and enabling user monitoring was added. This is useful in times when the user would like to use other functionalities of IntelliWeb, however does not want his or her browsing patterns monitored.

### 6.5 Database

A Microsoft SQL Server database was implemented in order to store user data. A user’s account information is used to authenticate them to the IntelliWeb tool. Users are referenced by their username and unique ID, which is then used to identify a user’s interest profiles, sessions and access patterns, saved RSS news feeds and bookmarks. The database schema is shown below.
A separate class was created to call stored procedures within the database. The use of stored procedures also warrants faster execution when compared to running individual SQL statements as stored procedures are compiled on the server when created. Furthermore, this supports modularity. If changes to the database are required, the application code does not have to be modified.

In situations where bookmarks and shortcuts were saved to the database and then immediately retrieved through a database query (such as showing the user their current bookmarks on adding a new bookmark to their profile), it was found that the latest additions to the database were not retrieved. On further investigation, it became apparent that the database was simply being queried too quickly in order to retrieve the recently added information. Therefore, to enable recently added information to be immediately retrieved, a separate thread was used to input the designated information to the database, sleep for a diminutive amount of time and then query the database as required. This time delay ensured the latest inputs to the database were retrieved, with the delay so short that the user remained unaware of its occurrence.

6.6 User interface

6.6.1 Client application

The user interface was implemented as below. The design and layout of the user interface was intentionally similar to that of a Microsoft Windows application, affording familiarity to the user and ease of use.

The use of tabs separates the various functions of the system. The functions that each tab offers are self evident in the tab titling. All inputs required from the user are clearly identified, to ensure effortless use and so that the user is constantly in control and aware of what is occurring.
6.6.2 User privacy
To ensure user privacy, authentication is required before the IntelliWeb system can be used. A username and password mechanism is used to identify users, and is necessary before the main functionalities of the system can be accessed. Users are able to create an account through registration. This method of identification allows unique interest profiles and access patterns to be determined for individual users. An invalid login attempt is confronted with a message box notifying the user of an unsuccessful login.

6.6.3 Suggestion alerts
Recommendations are a crucial aspect of the implementation of the IntelliWeb system. Although recommendations offered by the system are displayed in the client application...
window itself, the user would have to navigate back to the IntelliWeb user interface in order to view these recommendations. This does not follow the specifications regarding recommendations set in section 3, which states that new document recommendations, bookmarks and shortcuts should be made in unobtrusive and evocative manners, and not interrupt a user’s web browsing experience.

Some users may find tracking back to the IntelliWeb user interface for recommendations an interruption to their web browsing experience. A supplementary recommendation strategy was needed to unobtrusively notify users of recommendations.

Inspiration was found from Yahoo! Messenger [YAH], a program used by millions of web users worldwide. Yahoo! Messenger uses small pop-up windows in the bottom right corner of the user’s screen to alert the user of important events. A detailed tutorial of implementing such pop-ups is given by Taskbar Notify [POP].

A similar mechanism was implemented by the IntelliWeb application to notify users of recommendations. By implementing recommendations as fading pop-up windows, these suggestions can easily be ignored by the user or instantly closed. However, if they are of benefit to the user, the user can choose to click on the pop-up to navigate to the new page. This makes web browsing more effective for the user, and interruption free.

In order to allow the user to maintain control at all times, users are able to enable and disable this alert feature.

6.6.4 Bookmark collection

The bookmark collection tool, which was implemented as an extension to the IntelliWeb system, was designed in a similar manner to the IntelliWeb client application to ensure consistency of design. Help in the form of messages is given to the user throughout its use to ensure that the user is constantly in control and aware of what is occurring.
6.7 IntelliWeb overview

The use case diagram below depicts an overview of the methodology the system undertakes in providing the user with the functionalities it affords.

Figure 30 – Use case diagram of the IntelliWeb system
7 Project Management

This chapter identifies the project management procedures put in place to ensure the requirements of the IntelliWeb project were satisfied in good stead.

Project management is the discipline of defining standards and achieving targets whilst optimising the use of resources, which in this case is predominantly time, over the course of a project. Good project management is essential for the suave running of a project and to ensure the requirements of a project are satisfied in good stead.

7.1 Project security
There are a number of problems associated with the Internet [Dar06], which if not protected against can alter, monitor, or even damage a computer’s data. These ills include viruses, adware, spyware and other malicious software programs.

However, this is not the only security concern. In order to ensure data was not lost or destroyed through viruses, accidents, physical disasters or any other means, several security procedures were followed and are documented below:

1. Backups of all required data were taken. This included the IntelliWeb software code, metadata, project report and all other associated code, libraries and documentation.

2. Backups were made to another hard disk on the same computer, and to removable media, including DVD-R discs and a removable USB disk drive. This enabled the removal of the data copy to a safe location away from the original. A further electronic backup was also stored on a secure server accessible only through specific private information.

3. Backups were taken daily through a routine backup system.

4. Periodical reviews of the backup data were conducted in order to ensure the correct data was being copied and could be restored.

These steps were conducted to ensure that any loss of data was easy to recover from.

7.2 Documentation
There are a number of documents which must be created in order to support the IntelliWeb system. These include help guides on how to use the IntelliWeb system, and a project report documenting the design of the IntelliWeb agent from start to completion.

7.2.1 IntelliWeb documentation
It is essential help documentation is provided for the final IntelliWeb system. This acts as a user’s guide to the final software product, containing detailed information on how to use the system. It must also include a step-by-step guide for each of the systems function and a troubleshooting guide.

As the IntelliWeb tool is being tested by a group of BETA testers of varying levels of experience, it is essential that the supporting IntelliWeb document be clear, concise and easy to read and follow.

7.2.2 Project documentation
A full project report of the project must be provided, containing:

- Background research on the topic of intelligent agents and topics concerning the Internet, web users and related paradigms.
- A specification of how the system should be designed, supported by the research conducted.

- How the system is designed, highlighting the key decisions and technical challenges involved in their implementation.

- How the IntelliWeb system was implemented, with an overview of the system and its various components, and the problems found during their development.

- Detailed discussions on how the system will be tested, and by what features the IntelliWeb system will be judged upon.

- An evaluation based on the information gathered during the testing of the IntelliWeb system, and a conclusion discussing improvements and areas for future work given the current system.

### 7.3 Time management

In order to ensure the final product was completed in accordance with the given deadline, the IntelliWeb design was modularised. This allowed the establishment of milestones to be accomplished over given time periods. The Gantt chart below depicts the timing of activities over the time period given to the project.

Although not prescriptive, with the risk associated with the IntelliWeb design due to the new techniques and novel approaches discussed in section 4, the use of a Gantt chart allowed the analysis of the time available to further plan the development of the IntelliWeb tool.

Furthermore, with the use of a pictorial time management aid, it is evident upon reading of the modules which must be completed before further activities can take place.
Figure 31 – Gantt chart depicting the time management of the IntelliWeb project
8 Testing

This chapter details how the implementation of the IntelliWeb system was tested. The analysis and evaluation of these testing procedures is conducted in the proceeding chapter.

After implementation, an extensive range of subjective and objective tests were undertaken to evaluate the IntelliWeb system and the usability of the final product.

The testing of the system was conducted independently by a group of fifty individuals registered as BETA testers for Sitefinders Net Ltd (SFN), an e-commerce company based in Coventry, West Midlands. The individuals all opted-in to act as testers of new products with SFN, and range from novice to advanced Internet experience to ensure fairness.

As well as conducting a series of tests documented below, each user was given an online questionnaire to complete concerning the IntelliWeb system. This questionnaire was used to evaluate the effectiveness of recommendations and the usability of the system. This questionnaire can be seen in appendix B.

The testing of the final system can be split into several distinct cases:

1. Testing the contextual web search.
2. Testing the effectiveness of recommendations; of new documents, shortcuts, bookmarks and personalised news.
3. Testing the usability of the system.

Each of these test components is discussed in greater detail below.

8.1 Effectiveness of contextual search

Precision and recall are two performance measures that have been used by many similarity systems in order to measure their competence.

\[
\text{precision} = \frac{\text{number of relevant documents retrieved}}{\text{total number of documents retrieved}}
\]

Equation 8 – Precision determination

\[
\text{recall} = \frac{\text{number of relevant documents retrieved}}{\text{total number of relevant documents retrieved}}
\]

Equation 9 – Recall determination

Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is difficult to measure the recall metric in this situation however, as this is the ratio of the number of relevant records retrieved to the total number of relevant records. This metric is used when conducting database searches. As we are unaware of the total number of relevant records available on the web for a given query, it is impossible to reasonably calculate this metric.

Testing the effectiveness of the contextual search function provided by IntelliWeb was calculated using the precision metric alone.

Firstly, the testers were invited to search the web for a term of their choice, using their favourite search engine. The precision of the results returned was calculated. The testers were again asked to search the web. However, this time using IntelliWeb, specifying a search
query and domain pair, or simply a search query. The precision of the results returned by IntelliWeb was then calculated.

The number of results retrieved by IntelliWeb was set to ten, the number of search results retrieved by most search engines by default. This allowed the fair comparison of the precision metric for both methodologies.

Testers were encouraged to use ambiguous search terms, for example elephant, jaguar, angel, etc. As these metrics are reliant on human judgement of relevance, the definition of relevance was stated to be that the ‘results expected for the search term were the results received’.

The metasearch facility offered by IntelliWeb can be considered a success if the precision offered by IntelliWeb is greater than that of Google, Yahoo! and MSN for the same search terms.

8.2 Effectiveness of recommendations
In a system such as IntelliWeb, a vital functionality is to provide relevant recommendations to a user. However, as this is a subjective assessment, it is difficult to evaluate its success in a quantitative manner. A further complication is that the usefulness of recommendations change over time, as the system develops a more accurate representation of the user's interests.

However, inspiration was sought from the evaluation procedure used to test the success of the WebMate smart web browsing system [Che98]. This involved asking each user to specify a goal state that they wished to achieve by the end of their browsing session; a topic area that the user wanted to investigate.

During their browsing sessions, users were requested to identify the number of recommendations given, and the number taken. The number of recommendations taken by the user out of the number of recommendations given was used to give an indication to the usefulness of the recommendations. At the end of their browsing sessions, users were asked to identify and discuss in an online questionnaire whether the recommendations offered to them were of relevance, and whether any anomalous recommendations were given.

8.2.1 Bookmark and shortcut analysis
In order to determine whether the bookmarks and shortcuts functions performed correctly, two test procedures were conducted.

The first test involved asking users to browse the web as they would normally. As this was conducted in a separate browsing session to the recommendations analysis, there was no goal state in mind. However, users were asked to frequent at least several web sites during their browsing session, including a current affa irs web site and web-mail; as this provided multiple impressions to the same site.

Users were then questioned as to whether the bookmarks suggested were those of sites that they had frequented most, and whether the bookmarks were of use. Shortcuts were also tested in this way, with users questioned as to whether shortcuts offered to them were of web sites frequently visited when following their browsing habits.

The second, objective method of testing the bookmarks and shortcuts facility was to test the system component manually. This was done by inputting a predefined set of simulated page accesses to determine whether bookmarks and forward browsing paths were correctly identified. The test procedure and outcome of this assessment is discussed and evaluated in the proceeding section of this report. A similar method was used to test the bookmark collection tool implemented as an extension to the IntelliWeb tool to estimate user interests.
8.2.2 Personalised news and information
In order to determine whether news and information was relevant to the user’s browsing interests. Users were asked to save a number of RSS feeds to their profiles, either from a list supplied or from personal preference. As users browsed the web, they were asked to identify and discuss in an online questionnaire whether the articles displayed to them were of interest, and whether any anomalous articles were displayed.

8.3 System usability testing
The usability of the system is crucial to ensure the system meets the needs of the user. Hence usability of the system is best tested by asking the subjects testing the system whether the system adhered to the usability concerns specified in section 3 of this report. A further more technical assessment of the application’s usability was also conducted as identified in 8.3.2.

8.3.1 End-user testing
A part of the testing procedure involved a questionnaire making use of the Likert scale [LIK]. This was used to question users as to the extent they agreed or disagreed with a statement regarding the system. This was part of the online questionnaire the IntelliWeb testers completed at the end of the testing session.

Rather than using technical jargon in questions to the subjects, the questions were phrased in normal language. The statements were assembled such that half of the statements were positive in tone (odd numbered questions), and the other half, negative in tone.

These were designed to address Nielsen’s usability heuristics [USE] and address the system functionalities. The questionnaire given was as follows:

1. I would use this system again
2. This system was difficult to use
3. The search results that were brought back were what I expected and relevant
4. I could not use the system without consulting technical documents
5. Recommendations that were made were relevant and useful
6. Recommendations were not subtle and unobtrusive
7. I can use the system without consulting technical documents
8. The system was too complicated to use
9. The systems functions were easy to remember
10. When local searches were made, the results were relevant and useful
11. Mistakes were easy to rectify
12. I was not in control of the system
13. I feel the system is more useful than search engines for searching the web
14. Errors were not obvious and easy to cause
15. I enjoyed using the system
16. When I followed recommendations suggested to me they were irrelevant and useless
17. Bookmarks suggested to me were sites that I frequently visited and are were useful
18. I was not given recommendations that I was interested in
19. I would recommend the system to a friend
20. The system was slow and expensive to run

Figure 32 – Likert scale questionnaire given to testers of the IntelliWeb system

Each user responded with a rating as follows:

<table>
<thead>
<tr>
<th>Rating</th>
<th>Strongly agree</th>
<th>Agree</th>
<th>Neutral</th>
<th>Disagree</th>
<th>Strongly Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Negative</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 33 – Likert scale ratings
When using this method, a higher overall score indicates greater usability and performance. As the questionnaire was conducted online, each user’s submitted questionnaire was stored and evaluated at the end of the testing session.

### 8.3.2 Nielsen’s Usability Heuristics

It is reminded that the specifications stipulate that Nielsen’s Ten Usability Heuristics [USE] must be adhered to. Therefore, in order to determine whether this specification was met, the usability of the application was assessed in accordance to each of the ten heuristics. The procedure and outcome of this assessment is discussed in the proceeding section of this report.
9 Evaluation

This chapter conducts a quantitative analysis of the IntelliWeb test procedures, as well as an evaluation of the usability of the system, based on end-user and technical testing measures.

As individual tests were conducted, testers were asked to complete the relevant sections of the online questionnaire given to them. The use of an online questionnaire allowed the easy aggregation and analysis of data. An analysis of the results of the aforementioned test procedures is conducted below.

9.1 Contextual search

A representative sample of ten of the fifty contextual search test results is shown below. With search engines returning ten results by default, IntelliWeb was configured to return ten results also to allow straight comparisons.

<table>
<thead>
<tr>
<th>Query</th>
<th>User Ability</th>
<th>Search Engine</th>
<th>IntelliWeb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Search Engine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>Precision</td>
<td>Precision</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>L/Q</td>
</tr>
<tr>
<td>1 Football</td>
<td>Low</td>
<td>Yahoo!</td>
<td>40%</td>
</tr>
<tr>
<td>2 Restaurant</td>
<td>Medium</td>
<td>MSN</td>
<td>50%</td>
</tr>
<tr>
<td>3 Bullring</td>
<td>High</td>
<td>MSN</td>
<td>20%</td>
</tr>
<tr>
<td>4 Credit cards</td>
<td>Medium</td>
<td>Google</td>
<td>60%</td>
</tr>
<tr>
<td>5 World cup</td>
<td>High</td>
<td>Google</td>
<td>50%</td>
</tr>
<tr>
<td>6 Tenant loan</td>
<td>Low</td>
<td>Yahoo</td>
<td>20%</td>
</tr>
<tr>
<td>7 Orange</td>
<td>Low</td>
<td>Google</td>
<td>30%</td>
</tr>
<tr>
<td>8 Brick</td>
<td>High</td>
<td>Yahoo</td>
<td>10%</td>
</tr>
<tr>
<td>9 Cars</td>
<td>Medium</td>
<td>MSN</td>
<td>20%</td>
</tr>
<tr>
<td>10 Elephant</td>
<td>High</td>
<td>Google</td>
<td>20%</td>
</tr>
</tbody>
</table>

Figure 34 – Sample of users and their contextual search results

On analysis of the results, many noteworthy points are raised with regard to system performance. The IntelliWeb system returned an average precision of 59%. This compares advantageously to the search engines’ average precision of 32%. Local search (signified by L in the table above) proved most successful, with a 40% increased precision performance over standard search engine search.

Furthermore, adding a domain to the query (signified by Q), improved the relevance of the results retrieved by 32%. This statistic provides evidence to suggest that, on average, the IntelliWeb system provides more relevant search results to a user than the three search engines combined. However, without further analysis into the results, it is impossible to state this explicitly.

Nevertheless, on analysis of the individual search engines performance to IntelliWeb, one observes that IntelliWeb most outperformed the MSN search engine, by 36.6% (MSN: 30%, IntelliWeb: 66.6%). IntelliWeb also offered a 23.3% precision increase of 53.3% compared to Yahoo! search engine’s 30%. However, when comparing IntelliWeb to Google, IntelliWeb only offered a 10% increase in precision of 53.3%. This suggests that the search engine used to search the web greatly affects the relevance of results retrieved.

9.2 Recommendations

New document recommendations were assessed by the number of recommendations deemed relevant by a user, and thus taken. On average, 61% of recommendations given to a web user were taken. This was also analysed when taking into consideration user experience levels. Just over 76% of web users with low experience took recommendations offered to
them, compared with 41% of experienced web users. This suggests that lesser experienced web users are more inclined to follow recommendations by a system, whereas those of higher experience levels are more rigid in their browsing.

All web users stated that the recommendations made to them were relevant to their interest topics, even if not perused. Furthermore, although no anomalous recommendations were recorded by the testers, some testers did comment that on first using the IntelliWeb system, the recommendations made were not entirely conducive to their browsing habits.

Such statements support the need for an interest estimation facility, as was implemented as an extension to the IntelliWeb tool. As all of the IntelliWeb testers were using machines that did not belong to them, the use of web browser bookmarks to estimate their interests was of no use. However this was tested technically as discussed below.

9.2.1 Bookmark and shortcuts

Recommendations of bookmarks and shortcuts (forward paths) to a user are determined by evaluating the access patterns of a user. A technical assessment was conducted by providing predefined access logs to the system and evaluating its results to determine any inherent flaws in the system implementation.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Log</td>
<td>EFGHFFE</td>
<td>EFGH</td>
<td>EFGH</td>
<td>EGPL</td>
</tr>
<tr>
<td></td>
<td>EFGEF</td>
<td>EGH</td>
<td>EFGH</td>
<td>EFGH</td>
</tr>
<tr>
<td></td>
<td>EFH</td>
<td>EH</td>
<td>EFH</td>
<td>EHFF</td>
</tr>
<tr>
<td></td>
<td>FG</td>
<td>FH</td>
<td>FGH</td>
<td>GH</td>
</tr>
<tr>
<td></td>
<td>E: 4</td>
<td>F: 4</td>
<td>G: 2</td>
<td>E: 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>H: 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E: 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 35 – Technical testing of the bookmark and shortcut functionality

Single items marked in red depict bookmarks, with item lists in blue depicting which shortcuts which would be recommended should the user access any of the items in the list. Items in grey depict those ignored due to being under the required minimum support level. The results confirm the outputs are correct. There are several aspects worth noting:

- Bookmarks are offered for those items accessed thrice or more.
- Shortcuts are offered if accessed thrice or more.
- The bookmarks offered in test 1 and 4 state that F occurred once less than it did. This is due to the detection of repeat refresh abuse.
- The forward path EF in test 1 is only counted four rather than five times, due to the repeat refresh abuse.

This concludes that the proposed algorithm determines bookmarks and shortcuts to the user as defined.
9.2.1.1 Bookmark collection for interest estimation

The motivation behind this test was to determine whether a user’s interests could reasonably be estimated from their bookmarks. This test involved the importing of bookmarks into the IntelliWeb application using a blank user profile. The interests determined by the system were written to an output file for easy analysis. A sample test output is shown below.

The results from this test illustrate that only the most recent bookmarks are used to determine a user’s interests. Documents are separated by the period of time in which they were saved, and the most recent months bookmarks are used to estimate the user’s interest. This approach has the effect of estimating users’ interests for instant personalisation of recommendations and news in the IntelliWeb tool.

9.2.2 Personalised news

Users were asked to comment on the personalised news feeds which were tailored to the users’ interest profiles. Web users of all experiences commented on the advantages of personalised news and information. Furthermore, this signifies how user personalisation is a requirement which must be developed in future systems.
However, several users stated that whilst using the IntelliWeb system initially, the articles displayed to the user were not completely relevant. Upon greater use of the IntelliWeb system, users confirmed that the articles displayed were of greater relevance to their interests. This supports the use of the interest estimation tool implemented as an extension to the IntelliWeb system.

### 9.3 Usability

#### 9.3.1 End-user testing

The final section of the online questionnaire involved users discussing how much they agreed with a statement made regarding the system. The results of this procedure are shown below, with the average Likert scale rating for each question also given.

<table>
<thead>
<tr>
<th>+/-</th>
<th>Question</th>
<th>Average Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1+</td>
<td>I would use this system again</td>
<td>4.2</td>
</tr>
<tr>
<td>2-</td>
<td>This system was difficult to use</td>
<td>4.4</td>
</tr>
<tr>
<td>3+</td>
<td>The search results that were brought back were what I expected and relevant</td>
<td>3.8</td>
</tr>
<tr>
<td>4-</td>
<td>I could not use the system without consulting technical documents</td>
<td>4.8</td>
</tr>
<tr>
<td>5+</td>
<td>Recommendations that were made were relevant and useful</td>
<td>3.4</td>
</tr>
<tr>
<td>6-</td>
<td>Recommendations were not subtle and unobtrusive</td>
<td>4.2</td>
</tr>
<tr>
<td>7+</td>
<td>I can use the system without consulting technical documents</td>
<td>4.8</td>
</tr>
<tr>
<td>8-</td>
<td>The system was too complicated to use</td>
<td>4.9</td>
</tr>
<tr>
<td>9+</td>
<td>The systems functions were easy to remember</td>
<td>3.3</td>
</tr>
<tr>
<td>10-</td>
<td>When local searches were made, the results were neither relevant nor useful</td>
<td>3.9</td>
</tr>
<tr>
<td>11+</td>
<td>Mistakes were easy to rectify</td>
<td>4.8</td>
</tr>
<tr>
<td>12-</td>
<td>I was not in control of the system</td>
<td>4.5</td>
</tr>
<tr>
<td>13+</td>
<td>I feel the system is more useful than search engines for searching the web</td>
<td>3.3</td>
</tr>
<tr>
<td>14-</td>
<td>Errors were not obvious and easy to cause</td>
<td>4.7</td>
</tr>
<tr>
<td>15+</td>
<td>I enjoyed using the system</td>
<td>4.1</td>
</tr>
<tr>
<td>16-</td>
<td>When I followed recommendations suggested to me they were irrelevant and useless</td>
<td>4.0</td>
</tr>
<tr>
<td>17+</td>
<td>Bookmarks suggested to me were sites that I frequently visited and are useful</td>
<td>3.1</td>
</tr>
<tr>
<td>18-</td>
<td>I was not given recommendations that I was interested in</td>
<td>4.1</td>
</tr>
<tr>
<td>19+</td>
<td>I would recommend the system to a friend</td>
<td>4.5</td>
</tr>
<tr>
<td>20-</td>
<td>The system was slow and expensive to run</td>
<td>4.3</td>
</tr>
</tbody>
</table>

**Figure 36 – Average Likert scale ratings for usability questionnaire**

The average Likert rating for the questionnaire was 83.1 out of a possible 100. A high final Likert rating indicates that users found the IntelliWeb system met adequate usability and performance standards. However, as these were not compared to identical tests conducted for other similar systems, it is difficult to directly contrast IntelliWeb to the current state of the art.

From the questionnaire, it is evident that recommendations of new documents were not always relevant, with an average rating of 3.4 out of 5 for question 5. The majority of the lower ratings for this question came from highly experienced web users. The usefulness of such functionality will greatly vary over time. This occurs as the system develops a more accurate, refined profile of interests and user access patterns. Therefore, unless the system is used over a long period of time, the results cannot be used as concrete evidence of its effectiveness.
Extensive use and testing is required to determine the true effectiveness of the system, and this was not possible in a days testing, as was the case. However, this does support the use of an interest estimation facility to help refine the user's interest profile for initial browsing.

Comments left by several testers did expose some interesting details. Over 80% of testers commented that they would like to use the system again. However, several testers indicated that although the IntelliWeb system was useful and effective in assisting web browsing, the fact that the application had to be launched as well as the web browser was cumbersome. This was mentioned by four of the fifty testers; who were all of high experience levels. These testers mentioned that a similar system implemented in a web browser toolbar would be beneficial.

Testers of high experience levels found the shortcuts functionality most effective, with low experience level web users stating that the recommendations offered by the IntelliWeb system were relevant and most useful. Low experience level users commented that they were not able to make full use of the shortcuts feature of the system, suggesting that more experienced web users have regular repeated access patterns; and less experienced web users can be assisted greatly in their web browsing and have highly influenceable browsing habits.

9.3.2 Nielsen’s Usability Heuristics

Accordance to each of Nielsen's ten heuristics [USE] is discussed below.
1. **Visibility of system status:** The status bar is split into two panels to display the current status of the system. The first indicates whether the system is connected to the system, and the second denotes what the system is currently doing. This allows the user to keep track of their status within the system.

2. **Match between the system and the real world:** The use of tabs mimics the concept of pages to differentiate between the different functionalities of the system.

3. **User control and freedom:** All user actions are reversible. Thus if a user selects a system function by mistake, they are able to reverse this action. Furthermore, terminal functions met with confirmation message boxes.

4. **Consistency and standards:** The terminology and use of colours and icons are kept consistent to facilitate easy user comprehension of system functionality.

5. **Error Prevention:** The modular design of the system allows the disabling of sections of the system when not applicable for use reduced the scope for errors to occur.

6. **Recognition rather than recall:** The use of colour schemes and icons act to denote functionalities rather textual descriptions. This design feature promotes recognition of colours and icons rather than recall of textual descriptions of system functionalities.

7. **Flexibility and efficiency of use:** Flexibility of the system allows users to conduct tasks in a variety of ways. For example, users can double click an element in a list or explicitly select the element and click the button in order to perform the same task.

8. **Aesthetic and minimalist design:** The system is not cluttered with excessive use of icons and buttons. Tabs are used to separate different functionalities. A simple list arrangement is used to model information. The layout of the system is not dissimilar to that of a standard Microsoft Windows application.

9. **Help users recognise, diagnose and recover from errors:** Error messages are informative, explaining the cause of the error and advice to prevent the error from reoccurring.

10. **Help and documentation:** A comprehensive user guide is provided, and is easily accessible through the menu.

Examples of accordance to each of Nielsen’s 10 Usability Heuristics allow one to plausibly assume that an interface with effective usability has been successfully implemented. This suggestion is complemented by the user feedback discussed formerly in 9.3.1.

### 9.4 Comparison with state of the art

Due to restrictions upon the availability of state-of-the-art systems, it is problematic to directly compare the performance of IntelliWeb against other systems currently in the public domain. However, it is possible to compare the architectural and technical aspects of the various systems.

IntelliWeb makes use of innovative approaches concerning document representation. Previous systems such as WebMate [Che98] and Amalthaea [Mou97] have all used the TF-IDF scheme to represent a document. A document's HTML META tags are solely used to infer document semantics. However, with many web sites 'cheating' the index - optimising their web site for specific, targeted terms illegitimately, it was essential to use more than META tags to determine a documents true meaning. Given the nature of the web, this is a feature which must be incorporated into any future intelligent web system.

IntelliWeb tackles this problem by using META tags, document content and heading styles to determine a document's meaning. The document content is checked against the META tags.
to determine whether the META tags are representative of the documents true meaning. HTML heading styles are also examined to confirm the semantics of a document. This allows documents to be represented more accurately than solely using META tags. Furthermore, with the use of normalised TF-IDF vectors, a fairer comparison between documents is achieved.

A pioneering approach to user interest modelling was also incorporated. Systems such as WebMate [Che98] only allow the formation of new interests if the documents being viewed overlap with previous user interests. Any document which does not overlap with the user’s current interests is treated as a random page access and disregarded. Although a document may be irrelevant; as for example is an online advertisement, it is still feasible a new user interest could be forming. This approach does not permit the formation of new, completely different interests. Furthermore, this technique does not mirror the wandering nature of human attention.

Through the modelling of documents using the normalised TF-IDF scheme, the extraction of document semantics is achievable. Meaningless and commonplace words are extracted from the document before parsing. The user interests are determined from the most frequent terms appearing in the documents being viewed by the user. It is then possible to determine whether new documents being accessed share any of the user’s current interests. If they do not, these terms are added to the user’s interest profile with low significance. Therefore, as the user continues their web browsing, should these terms begin to feature prominently in observed documents, the significance of these terms increase and begin to dominate the interest profile. This allows new interests to form, without the hindrance of requiring interests to overlap.

Furthermore, with an element of decay applied to all items within a user interest profile, random page accesses and interests which become peripheral deteriorate until eventually being removed. However, this feature also allows the resurgence of any previous user interests should they be rekindled. This feature was incorporated to mirror the deterioration of user interests, and again was not incorporated by the majority of current systems.

Recommendations are then offered based on the prominent terms within a user’s profile. These terms are used to perform a metasearch in order to suggest new documents of interest to the user. An additional functionality of news personalisation was also incorporated into the IntelliWeb design. This allows the personalisation of news and information in line with the user’s interests.

The results obtained from testing seem to indicate a successful implementation of the recommendation facility, most welcomed by lesser experienced web users. However, as recommendations are based purely on subjective judgement, it is impossible to comprehensively evaluate such a facility against other systems.

All of the systems currently available form a user profile over a period of time, starting from scratch. IntelliWeb also does this, however offers the extended functionality of estimating a user’s interests from their web browser bookmarks. These interests are then used to offer immediate recommendations and personalisation of news, and are refined as the user browses the web. Although this is a highly useful tool, it can be argued that web browser bookmarks are not the best way to judge one’s interests. The analysis of other information, such as personal documents, although a highly controversial method, may prove a better method of judging interests, and this could be investigated in further systems.

The novel facility of metasearch is offered by IntelliWeb. This feature has not been offered by any system to date. With a paradigm-shift being witnessed in how search engines are used by web users, it was essential to implement a facility which allowed users to query multiple search engines to determine contextually relevant and localised results to their queries. Users are able to query a plethora of search engines through one interface, with the addition of a query domain to ensure semantically relevant results. Localised results are retrieved through the autonomous calculation of the user’s current location. The user’s location and query domain are used to filter the results returned to ensure relevance.
Furthermore, before being displayed to the user, the search engine results returned undergo a strict ranking policy, as determined from the background research. Web sites that are currently unavailable are removed from the search engine results, and the notion of URL affordance is introduced, in order to remove spam web sites from the search engine results. This ensures greater confidence in the results returned; a key aspect of Web 2.0. The complex determination of document semantics through use of HTML META tags, document content and heading styles also allowed finer filtration of results.

The results obtained from testing seem to indicate a successful implementation of the metasearch facility. The metasearch facility offered by IntelliWeb outperformed the top three search engines, with an average 27% greater precision in results returned.

Systems such as Footprints [Wex97] make use of user access patterns to make recommendations. However, Footprints only presents recommendations for pages within the site being visited. Given the highly dynamic nature of web sites today with ever expanding content, a different approach was implemented in IntelliWeb. IntelliWeb offers a shortcuts facility, which offers users shortcuts to sites they frequent when on particular browsing access patterns.

A similar functionality to that offered by Alexa [ALE] could be added as an improvement to the system. Given the amount of data collected by IntelliWeb, it would be possible to offer collaborative recommendations based on other users’ browsing patterns. This is offered by Alexa, with a list of popular sites also frequented by users visiting a particular site. Such information would be useful to users who share common interests.

Letizia [Lie97] and PowerScout [Lie01] operate in modified Netscape Navigator web browsers. Netscape Navigator is no longer a popular web browser and hence these systems are unlikely to be used by many web users. IntelliWeb was designed to work without the use of a proxy server due to the difficulty in imposing proxies on many networks, especially networks in places of employment; as such networks commonly operate a prudent firewall [SOA]. Therefore, IntelliWeb was designed to work alongside Internet Explorer versions 4.0 onwards, encompassing over 75% (worst case) of the world’s web users. Although this is still a large target audience, the modification of IntelliWeb to work with all web browsers would be greatly welcomed.

Furthermore, due to the use of a .NET framework, IntelliWeb only runs on computers running the Microsoft Windows operating system. Although this counts for the majority of computer users, to increase the scope of IntelliWeb’s use, this could be modified to work across all operating systems. The use of a programming language such as Java [SUN] may prove useful in the implementation of such an improvement, although none of the current systems in the public domain make use of this language. Java is an extremely portable implementation language; although it does rely on having a Java Virtual Machine present on the operating machine.

In the context of this project, especially with the recurrent use of the web and hence any application that may be used alongside it, it is crucial that any implementation adhere to strict usability guidelines. The user interfaces of such systems are critical to ensure that users are both able and inclined to use the systems. This was a contributing factor to the use of a .NET architecture for the system; due to its familiar look and feel.

The evaluation of system usability, through the use of a Likert scale questionnaire to gauge tester opinions (discussed in section 9.3.1) and a more technical assessment (discussed in section 9.3.2) all indicate that the system meets the usability specifications discussed in the specification.

9.5 Evaluation summary

Analysis of the testing results indicates there is positive reason to believe the contextual search provides more relevant search results to a user. Furthermore, it is reasonable to
believe the user interest and access pattern modelling proposed in the project provides a more accurate representation of a user’s browsing habits.

Moreover, with the clear adoption of stringent usability guidelines and a positive response from the end user testing, there is a strong indication that the usefulness and effective of the IntelliWeb system exceeds a satisfactory level.
10 Conclusion

This chapter concludes the report, looking back at what has been accomplished and presents a basis for the opportunities of future work.

This report documents extensive research into the concept of a smart web browsing, with the comparison and evaluation of currently available systems. New techniques such as metasearching and information personalisation were introduced, as well as novel approaches to existing methods such as HTML parsing, user interest profiling and access pattern determination.

A final system, IntelliWeb, was implemented. IntelliWeb offered several major functionalities. These included metasearch with contextual and localised personalisation, recommendations of new documents of interest based on an individual’s interest profile and browsing patterns, and personalised news and information. Finally, a comprehensive testing procedure and intensive system evaluation was conducted to draw attention to the positive and negative contributions of the IntelliWeb tool.

Time and consideration was taken to ensure the design and implementation of a fully functional and usable end-user application. IntelliWeb was designed such that the addition of further components would be simple to integrate into the application. This is as a result of the software engineering approaches used during the implementation phase.

The outcome was a final product which was deemed more than adequate to be released into the public domain.

After the testing procedure, based on the success of the IntelliWeb tool, I was offered a contract to produce a similar product to the IntelliWeb system by SFN; for distribution to their half a million plus user base. The offer letter can be seen in appendix B.

10.1 Future work

IntelliWeb offers several novel functionalities, and when compared to previous systems, provides considerable architectural and conceptual advantages. However, there are many areas to be investigated further, and several different areas of future work can be identified as a result of the systems achievements.

10.1.1 Advanced user collaboration

With the client-server architecture of the IntelliWeb tool, the amalgamation of real-time user information is possible. The analysis of user browsing sessions and pages that users have explicitly marked as bookmarks could be used to deduce cross-user similarities to power further recommendations for other users. Further user data collection in the form of user ratings of pages or explicit feedback could also be used. This data could be used to determine the types of pages users find irrelevant or distasteful, in order to further refine recommendations for other users.

Business community web sites such as Alexa [ALE] could also be used to determine the most popular forward browsing paths taken by users when frequenting particular web sites. However the tool provided by Alexa is chargeable and only operates a one thousand lookups per day policy. With the use of an architecture allowing concurrent use, this limit will easily be exceeded; especially with web users visiting a variety of web pages in any given browsing sessions.

10.1.2 Lexical analysis

An approach that could improve the perception of information would be the lexical analysis of terms within a document. The incorporation of a lexical reference system such as WordNet [WOR] would allow inferences to be deduced on the semantic meaning of various terms by
looking at the other terms present in the same document. If an estimation of the definition of terms based on the context of the document can be determined, the similarity between documents containing similar semantic terms could be adopted. This approach allows the similarity between documents to be calculated more accurately than the standard keyword extraction approach. This is similar to the methodology the metasearch engine undertakes when a user specifies a domain for their query.

10.1.3 Web service implementation
IntelliWeb was produced as a client side user application. Yet, given the nature of the IntelliWeb application, several advantages could be achieved through implementing the system as a web service. There would be a new series of technical challenges to overcome. However a web based architecture would afford accessibility to a wider audience and multi-browser support would become implicit depending on the language used to design the system. This would provide a more accessible and usable end product.

10.1.4 Multi-Browser support
The IntelliWeb tool currently supports Internet Explorer 4.0+. Taking this into consideration, at best the population of web users with an IntelliWeb supported web browser stands at 82.75%, and at worst 75.5% of web users [BRO]. Therefore, to open the IntelliWeb to a greater user base, the system could be designed to function with other web browsers. This would encourage greater usage.

10.1.5 Experience dependence
During testing, it was evident that novice web users and experienced web users found different types of recommendations useful. Therefore, recommendations of documents of interest to a user could to some degree be based on the experience level of the user. The experience of a user could be autonomously determined by the time users spend on particular web sites, the types of web sites visited by a user, or the breadth of user interests.

10.1.6 Browser integration
During the testing phase of the IntelliWeb tool, many testers spoke of how a similar tool encompassed into a browser toolbar would be of even greater benefit to a web user. Many search engines and service providers provide toolbars which offer their services to users at the click of a button [EBA, GOO, MSN, YHO, ALE, ALT]. The reasoning behind this is that a browser toolbar does not need to be loaded up before using the web as IntelliWeb currently does, as it is incorporated into the browser on launch. Therefore, if the IntelliWeb functionalities were able to be focused into a browser toolbar, this would cater for a greater number of web users.

This improvement is the main architectural modification requested by SFN in their contract to produce a similar product to the IntelliWeb system.

The only disadvantage of such an approach is that browser toolbars are often browser specific. Therefore unless a plethora of toolbars are produced to cater for each of the main available web browsers in the public domain, the toolbar produced will have to be implemented to work with the most popular web browser.
11 Bibliography

11.1 Publications

Robert Armstrong, Dayne Freitag, Thorsten Joachims and Tom Mitchell
1997

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

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

[Bru99] Overview and Preview Tools For Navigating the World-Wide Web
Benjamin D. Brunk
1999

[Cha03] Personalized and Focused Web Spiders
Michael Chau and Hsinchum Chen
2003

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

[Chk03] Distributed Hypertext Resource Discovery Through Examples
Soumen Chakrabarti, Martin H. van den Berg and Byron E. Dom
1999

[Dar06] Economics of the Internet
John Darlington, Imperial College London
2006

[Dav02] Predicting Web Actions from HTML Content
Brian Davison
2002

[Coo97] Web Mining: Information and Pattern Discovery on the WWW
R Cooley, B Mobasher and J Srivastava
1997

[Eir03] SEWeP: Using Site Semantics and a Taxonomy to Enhance the Web Personalization Process
M. Eirinaki, M. Vazirgiannis and I. Varlamis
2003

[Fal05] Search Engine Users
Deborah Fallows, PhD, Senior Research Fellow
2005

[Fan04] A Review of Web Mining Research
Fan Zhao; University of Nebraska, Lincoln
2004
[Fen03] Client Side Monitoring for Web Mining
Kurt D Fenstermacher and Mark Ginsburg
2003

[Fin01] Placing Search in Context: The Concept Revisited
Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias and Ehud Rivlin
2001

[Fla02] Naive Bayesian Classification of Structured Data
Peter Flach and Nicolas Lachiche
2002

[Han00] Data Mining: Concepts and Techniques
Jiawei Han and Micheline Kamber
2000

[Her96] Intelligent Software Agents on the Internet: an inventory of currently
offered functionality in the information society and a prediction of
developments
Björn Hermans
1996

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

[Lie95] Letizia: Agent that assists in web browsing
Henry Lieberman
1995

[Lie97] Autonomous Interface Agents
Henry Lieberman
1997

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

[Rus03] Artificial Intelligence: A Modern Approach
S Russell and P Norvig
2003

[Lya00] Web Usability Patterns
Fernando Lyardet and Gustavo Rossi
2000

[Men99] Towards a Highly-Scalable and Effective Metasearch Engine
Weiyi Meng, Zonghuan Wu, Clement Yu and Zhuogang Li
1999

[Mou97] Amalthaea: Information Filtering & Discovery using Multiagent Evolving
Systems
Alexandros Moukas
1997

[Nan01] Mining Patterns from Graph Traversals Data and Knowledge Eng. (DKE),
Alexandros Nanopoulos and Yannis Manolopoulos
2001
Towards Adaptive Web Sites: Conceptual Framework and Case Study
Mike Perkowitz and Oren Etzioni
1999

An Algorithm for Suffix Stripping
M. Porter
1980

Data Mining – Matching and the Index
Martin Russell; University of Birmingham
2004

Latent Semantic Indexing
Martin Russell; University of Birmingham
2005

Introduction to Modern Information Retrieval
G Salton and M J McGill
1983

Adaptivity through Unobtrusive Learning
Ingo Schwab and Alfred Kobsa
2002

An Introduction to Neural Networks
Prof Leslie Smith; University of Stirling
1996

Agent-assisted Internet Browsing
Gabriel Somlo and Adele E. Howe
1999

11.2 Web sites

Alexa
http://www.alexa.com

AltaVista
http://www.altavista.com

Amazon
http://www.amazon.com

Search engine users stop at page three
http://news.bbc.co.uk/1/hi/technology/4900742.stm

Benchmark tests -.NET 2.0 vs. Java 1.5

Browser Statistics
http://www.upsdell.com/BrowserNews/stat.htm

BBC World News
http://www.bbc.co.uk/news

Relational databases
http://www.agiledata.org/essays/relationalDatabases.html
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>[EBA]</td>
<td>eBay</td>
<td><a href="http://www.ebay.com">http://www.ebay.com</a></td>
</tr>
<tr>
<td>MSQ</td>
<td>Microsoft SQL</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>POP</td>
<td>Taskbar Notify – MSN style pop-ups</td>
<td></td>
</tr>
<tr>
<td>RSS</td>
<td>What is RSS?</td>
<td></td>
</tr>
<tr>
<td>SEA</td>
<td>Search Engine Feature Chart</td>
<td></td>
</tr>
<tr>
<td>SEO</td>
<td>Differences between Search Engines and Web Directories</td>
<td></td>
</tr>
<tr>
<td>SER</td>
<td>Optimum Keyword Density – How it effects search engine rankings</td>
<td></td>
</tr>
<tr>
<td>SOA</td>
<td>SOAP – Benefits</td>
<td></td>
</tr>
<tr>
<td>SMA</td>
<td>Smart tools for Smart Searching</td>
<td></td>
</tr>
<tr>
<td>STO</td>
<td>Stopword List Removal</td>
<td></td>
</tr>
<tr>
<td>SUN</td>
<td>Sun Microsystems Java</td>
<td></td>
</tr>
<tr>
<td>TXT</td>
<td>Text Retrieval Conference</td>
<td></td>
</tr>
<tr>
<td>TRO</td>
<td>User interface – How many clicks?</td>
<td></td>
</tr>
<tr>
<td>USA</td>
<td>Usability 101</td>
<td></td>
</tr>
<tr>
<td>USE</td>
<td>Nielsen’s Usability Heuristics</td>
<td></td>
</tr>
<tr>
<td>USR</td>
<td>Usability and User Experience Testing</td>
<td></td>
</tr>
<tr>
<td>WOR</td>
<td>WordNet</td>
<td></td>
</tr>
<tr>
<td>WSE</td>
<td>Web Services and Other Distributed Technologies Developer Center: Web Services Enhancements (WSE)</td>
<td></td>
</tr>
<tr>
<td>YEL</td>
<td>Yell.com</td>
<td></td>
</tr>
</tbody>
</table>
[YAH] Yahoo! Messenger
http://www.yahoo.com/messenger

[YAP] Yahoo! API
http://developer.yahoo.com/

[YHO] Yahoo!
http://www.yahoo.com
Appendix A – IntelliWeb User Guide

This is the user guide, detailing the use of IntelliWeb.

IntelliWeb comprises of two applications:

- The IntelliWeb application (IntelliWeb.exe)
- The Bookmark collection tool (IW_CollectBookmarks.exe)
- A further IntelliWeb user guide document complements the IntelliWeb application (IW_UserGuide.doc)

A.1 Setup

A.1.1 IntelliWeb setup

i. To setup IntelliWeb, simply copy the IntelliWeb application and Bookmark collection tool into a directory of your choice, for example C:\IntelliWeb.

ii. No further configuration of your computer or browser is required.

iii. To launch the IntelliWeb application, open the IntelliWeb.exe file.

A.2 Using IntelliWeb

A.2.1 Registration and login

On the launch of the IntelliWeb application, you will be required to enter your username and password in the login screen. You must set up a personalised account so that personalised recommendations of new documents, bookmarks, shortcuts and personalised up-to-date news can be given to you as you browse. Enter your username and password and click the button labelled ‘login’ to use the application.

If this is your first time using IntelliWeb and you do not have a username and password set up, click on ‘click here to register’ to set up an IntelliWeb account.

When the register screen appears, type in your desired username and password and click the button labelled ‘register’.

If your desired username and password has already been taken, you will be informed and requested to enter a new username and password combination until a username that has not already been allocated is entered. Once an account has successfully been created, you will be taken to the IntelliWeb main application.

Make sure you remember your username and password as this is necessary to retrieve your personalised information.
A.2.2 IntelliWeb overview
After having successfully logged in, you will see the IntelliWeb screen before you. The IntelliWeb tool has a number of functions to assist you in browsing the Internet as well as tailoring the web to suit your needs.

There are four tabs to the IntelliWeb tool.

A.2.3 Search
The search tab offers you the ability to search the web contextually and locally.

- By adding a **domain** to your query, the chance of retrieving irrelevant results is dramatically reduced.

- By selecting **local search**, only results that are geographically local to you will be retrieved. This is useful when searching for services in your area, such as estate agents or plumbers.

- If a **quick search** is selected, the system only analyses the META tags, URL and TITLE when ranking the results retrieved. If quick search is not selected, the document content is also analysed. Although this takes slightly longer than a quick search, it is often more precise.

- The **search options** button opens a screen which lets you specify how many results you’d like to be returned to your search, and lets you set your location. Your location is automatically detected, however if you would like to change it, you can do it here. It
also lets you set whether you want pop-up notifications of documents of relevance to be displayed to you.

- When you have made a search, when the results are retrieved, a screen similar to this will be shown:

When you make a search, the results returned will be ranked.

When you click on a search result, a description of the search result will be shown in the 'Result Summary', displaying the web address of the search result and a weighting representing the value of merit associated to the search result.

- Once you click on 'Go to Result', the web page will open in a new browser window. Recommendations of relevant documents of interest will be determined if this option is not switched off in the search options.

A.2.4 Suggestions
When you visit a web site, IntelliWeb will recommend documents of relevance to you based on your user profile.

The best ranked suggestion will be displayed in a pop-up fashion on the right hand side of your screen just above the task bar. This will happen each time you visit a new web site unless you disable pop-up notifications.
The full list of suggestions will be displayed under the suggestions tab of the IntelliWeb tool.

Suggestions are made by querying Google, MSN and Yahoo! search engines with your current interests.

If you would like suggestions recommended to you as you browse, enter the address of the web site you would like to visit and press ‘Go’.

Thereon recommendations will be given to you unless otherwise stated.

The full list of suggestions retrieved is displayed here, ordered by rank.

The details, address and weighting of the suggestion is given here.

Click here to visit the selected web site.

- If you choose to visit a suggestion, the suggestion will open up in a new browser window if none are open; else will open in the currently open browser window.

- The weighting of the suggestion is an estimation of the similarity of this page to your interests.

- The suggestion summary gives you a brief overview of the suggestion link.

A.2.5 Shortcuts
The shortcuts tab lets you view your currently saved bookmarks, lets you manage your bookmarks and view shortcuts and bookmarks that may be of benefit to you whilst you browse the web.

As you browse, bookmarks of sites that you frequent will be offered to you, as well as links to sites that you often visit; these are known as shortcuts.

Shortcuts and bookmarks will both be offered as pop-up suggestions as well as under the shortcuts tab.
The suggestions tab will appear similarly to this:

- If you choose to visit a URL stored in your bookmarks, it will open in a new browser window if none are open; else it will open in the currently open browser window.
If you choose to **add the current URL** to your bookmarks, the web address of the document you are currently looking at in your web browser will be added to your bookmarks. If you do not have a web browser window open, you will be displayed a message informing you that this is the case.

### A.2.6 My News

This tab displays the latest personalised news to you, based on your current interests as determined by your web browsing habits.

![IntelliWeb Tool](image.png)

- **To add an RSS bookmark**, add it in the box provided and click this button.
- **To remove an RSS feed**, click the RSS feed to select it. The button labelled 'Add' will change to 'Del'. Click this button to remove the RSS feed.
- A list of the personalised news feeds is shown here. Different colours denote different RSS feeds.
- **If you click a news article**, a description of the article will appear.
- **Click on this button** to edit the number of news articles displayed.
- **Click this button** to view the news article's web page.

### A.2.7 Further options

- The ‘File’ icon in the taskbar allows you to edit the search options, edit the news options, import your bookmarks, and exit the application.
- The ‘Help’ taskbar icon allows you to view the help guide and find out about the application.
A.3 IntelliWeb add-ons

IntelliWeb is able to give you instant recommendations and personalised news with the use of the Collect Bookmarks tool, which exports your bookmarks from your favourite web browser and imports them into IntelliWeb in order to refine your interest profile.

A.3.1 Export your bookmarks from your web browser

To export your bookmarks from your web browser:

2. Select Export Favourites.
3. Follow the instructions until your bookmarks have been exported.

A.3.2 Convert bookmarks into readable XML format

To convert your bookmarks into a format IntelliWeb can handle:

1. Launch the Bookmark collection tool.
2. Click on the ‘Open File’ button, and select the exported bookmark file you exported from your web browser.
3. Choose a folder to save the XML file in. If you do not choose a directory, the file will be saved directly to your ‘C:\’ drive.
4. Press the ‘Collect my Bookmarks’ button to transfer your bookmarks into XML. The new file will be called IWTBookmarks.xml and will be saved in the directory you selected.

A.3.3 Import XML bookmarks into IntelliWeb

To import your bookmarks into IntelliWeb to help refine your interest profile:

1. Click File > Bookmarks > Import XML bookmarks

2. IntelliWeb will automatically parse this document and use the most recent bookmarks to help refine your interest profile.

By doing this, your interest profile can immediately be determined in order to give more accurate recommendations and personalised up-to-date news through the My News tab.
Appendix B – Additional Testing Information

B.1 Questionnaire
The questionnaire which was completed online by each of the testers during the period of IntelliWeb testing is shown below.

B.1.1 Questionnaire - section 1

B.1.2 Questionnaire - section 2
B.1.3 Questionnaire - section 3

B.1.4 Questionnaire - section 4
### B.1.5 Questionnaire - section 5

**Usability - 1/2**

Please answer the following questions, giving each question a mark from 1 to 5. The marking scheme implies the following:

1: Strongly agree, 2: Agree, 3: Neutral, 4: Disagree, 5: Strongly disagree

<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I would use the system again</td>
<td></td>
</tr>
<tr>
<td>2. This system was difficult to use</td>
<td></td>
</tr>
<tr>
<td>3. The search results that were brought back were what I expected and relevant</td>
<td></td>
</tr>
<tr>
<td>4. I could not use the system without consulting technical documents</td>
<td></td>
</tr>
<tr>
<td>5. Recommendations that were made were relevant and useful</td>
<td></td>
</tr>
<tr>
<td>6. Recommendations were not subtle and unobtrusive</td>
<td></td>
</tr>
<tr>
<td>7. I can use the system without consulting technical documents</td>
<td></td>
</tr>
<tr>
<td>8. The system was too complicated to use</td>
<td></td>
</tr>
<tr>
<td>9. The systems functions were easy to remember</td>
<td></td>
</tr>
<tr>
<td>10. When local searches were made, the results were relevant and useful</td>
<td></td>
</tr>
</tbody>
</table>

Next >>

### B.1.6 Questionnaire - section 6

**Usability - 2/2**

Please answer the following questions, giving each question a mark from 1 to 5. The marking scheme implies the following:

1: Strongly agree, 2: Agree, 3: Neutral, 4: Disagree, 5: Strongly disagree

<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>11. Mistakes were easy to rectify</td>
<td></td>
</tr>
<tr>
<td>12. I was not in control of the system</td>
<td></td>
</tr>
<tr>
<td>13. I feel the system is more useful than search engines for searching the web</td>
<td></td>
</tr>
<tr>
<td>14. Errors were not obvious and easy to cause</td>
<td></td>
</tr>
<tr>
<td>15. I enjoyed using the system</td>
<td></td>
</tr>
<tr>
<td>16. When I followed recommendations suggested to me they were instructive and useful</td>
<td></td>
</tr>
<tr>
<td>17. Suggestions suggested to me were sites that I frequently visited and were useful</td>
<td></td>
</tr>
<tr>
<td>18. I was not given recommendations that I was interested in</td>
<td></td>
</tr>
<tr>
<td>19. I would recommend the system to a friend</td>
<td></td>
</tr>
<tr>
<td>20. The system was slow and expensive to run</td>
<td></td>
</tr>
</tbody>
</table>

Thank you for your time in testing the IntelliWeb system.

Next >>

### B.2 Indication of testing success

The letter received from SFN after testing, offering me a contract to develop a similar application to IntelliWeb immediately follows this page.