Colombus: Providing Personalized Recommendations for Drifting User Interests

Ioannis Psarras

March 26, 2009

Department of Computing Science

Faculty of Computing Science, Mathematics and Statistics

University of Glasgow

UNIVERSITY
of
GLASGOW

To my family

Abstract

The query formulationg process if often a problematic activity due to the cognitive load that it imposes to users. This issue is further amplified by the uncertainty of searchers with regards to their searching needs and their lack of training on effective searching techniques. Also, given the tremendous growth of the world wide web, the amount of imformation users find during their daily search episodes is often overwhelming. Unfortunatelly, web search engines do not follow the trends and advancements in this area, while real personalization features have yet to appear. As a result, keeping up-to-date with recent information about our personal interests is a time-consuming task. Also, often these information requirements change by sliding into new topics. In this case, the rate of change can be sudden and abrupt, or more gradual.

Taking into account all these aspects, we believe that an information assistant, a profile-aware tool capable of adapting to users' evolving needs and aiding them to keep track of their personal data, can greatly help them in this endeavor. Information gathering from a combination of explicit and implicit feedback could allow such systems to detect their search requirements and present additional information, with the least possible effort from them.

In this paper, we describe the design, development and evaluation of Colombus, a system aiming to meet individual needs of the searchers. The system's goal is to pro-actively fetch and present relevant, high quality documents on regular basis. Based entirely on implicit feedback gathering, our system concentrates on detecting drifts in user interests and accomodate them effectively in their profiles with no additional interaction from their side.

Current methodologies in information retrieval do not support the evaluation of such systems and techniques. Lab-based experiments can be carried out in large batches but their accuracy often questione. On the other hand, user studies are much more accurate, but setting up a user base for large-scale experiments is often not feasible. We have designed a hybrid evaluation methodology that combines large sets of lab experiments based on searcher simulations together with user experiments, where fifteen searchers used the system regularly for 15 days. At the first stage, the simulation experiments were aiming at

tuning Colombus, while the various component evaluation and results gathering was carried out at the second stage, throughout the user study. A baseline system was also employed in order to make a direct comparison of Colombus against a current web search engine. The evaluation results illustrate that the Personalized Information Assistant is effective in capturing and satisfying users' evolving information needs and providing additional information on their behalf.

Contents

I Introduction **1**

1 Introduction **2**

1.1 Introduction 2

1.2 Thesis Layout 5

2 Background and Motivation **6**

2.1 Relevance Feedback 6

2.1.1 Explicit Ratings 7

2.1.2 Implicit Ratings 8

2.1.2.1 Sources of Implicit Ratings 9

2.2 Personalization System 11

2.2.1 Personalization through Recommendations 12

2.2.2 Personalization through Query Expansion 14

2.2.3 Personalization through Result Re-ranking 15

2.3 Evaluating Personalization Systems 15

II User Profiling **20**

3 User Profiling **21**

3.1 Dynamics of Search Needs . . . 21

3.1.1 Long-term vs. temporal search needs . . . 21

3.2 Detecting and Adapting to Drifting Concepts . . . 23

3.2.1 Types of Topic Drifts . . . 24

3.2.2 Detecting Topic Drifts . . . 24

3.2.3 Handling Topic Drift . . . 25

3.2.3.1 Time window . . . 25

3.2.3.2 Ensemble Classifiers . . . 27

3.2.3.3 Decay functions, Forgetting Factors and Half-Life . . . 28

3.3 Topic Overlapping . . . 30

4 Profile Learning and Representation **31**

4.1 Profile Learning . . . 31

4.2 Profile Representation . . . 33

4.2.1 Weighted Semantic Networks . . . 35

4.2.2 Binary Keyword Vector Approach . . . 35

4.2.3 Weighted Keyword Vector Approach . . . 36

III Colombus Architecture **37**

5 User Interface **38**

5.1 Interaction with the User Interface . . . 39

5.2 Registration / Login Interface 40
5.3 Search Interface 40
5.3.1 Document Summarization 41
5.4 Recommendation Portal 42
5.5 Document Bookmark 43
5.6 User Profile Management 44

6 System Design & Implementation **45**

6.1 Introduction 45
6.2 Process Overview 46
6.3 Search Module 47
6.4 Relevance Feedback 48
6.5 Term Selection 48
6.5.1 Voting vs. Non-voting schemes 49
6.5.2 Binary term weighting 51
6.5.3 Term frequency (TF) 51
6.5.4 Term Frequency - Inverted Document Frequency (TF-IDF) 52
6.5.5 WPQ 52
6.5.6 Jeffrey's Conditioning 53
6.6 Profile Learning 54
6.6.1 Clustering Algorithm 56
6.7 Recommendation Process 56
6.8 Implementation 57

6.8.1 High-Level Architecture . . . 57
6.8.2 A Three-Tier Model . . . 57
6.8.3 JAVA Servlet Technology . . . 58
6.8.4 Profile Persistence . . . 59

IV System Evaluation 61

7 Simulation 63

7.1 Introduction . . . 63
7.2 System, Collection and Topics . . . 63
7.3 Designing Searcher Simulations . . . 64
7.3.1 User Patience . . . 65
7.3.2 User Experience . . . 65
7.3.3 Previous Knowledge . . . 65
7.3.4 Creating a Searcher Simulation Model . . . 66
7.4 Simulation Components . . . 67
7.5 Experimental procedure . . . 68
7.6 Evaluation Measures . . . 68
7.7 Simulation Scenarios . . . 70
7.7.1 Scenario 1 . . . 70
7.7.2 Scenario 2 . . . 71
7.7.3 Scenario 3 . . . 73
7.8 Discussion . . . 75
7.9 Simulation Statistics . . . 76

8 Simulation Result Analysis **77**

8.1 Scenario 1 . . . 77

8.1.1 Evaluating Term Weighting Schemes . . . 77

8.1.2 Evaluating Clustering/Document Threshold Configurations . . . 86

8.2 Scenario 2 . . . 90

8.2.1 Evaluating Term Weighting Schemes . . . 90

8.2.2 Evaluating Clustering/Document Threshold Configurations . . . 96

8.3 Scenario 3 . . . 97

8.3.1 Evaluating Term Weighting Schemes . . . 97

8.3.2 Evaluating Clustering/Document Threshold Configurations . . . 100

8.4 Discussion . . . 102

9 User Experiments **106**

9.1 Introduction . . . 106

9.2 Experimental Subjects . . . 107

9.3 Experimental Tasks . . . 108

9.4 Experimental Systems . . . 109

9.4.1 Colombus . . . 110

9.4.2 Baseline (SearchIT) . . . 111

9.5 Experimental Hypothesis . . . 111

9.6 Experimental Procedure . . . 112

9.7 Data Collection . . . 113

9.7.1 Questionnaires . . . 113

9.7.2 System Logging . . . 113
9.7.2.1 Profile-related questions . . . 113
9.7.2.2 Task-related questions . . . 114
9.7.2.3 General questions . . . 114

10 Results and Analysis **115**

10.1 Profile Capturing . . . 115
10.2 Recommendation Performance . . . 120
10.3 System Preference . . . 121
10.4 Evaluation Hypothesis Analysis . . . 126
10.5 User Comments . . . 130

V Conclusion **131**

11 Conclusion **133**

11.1 Implicit Feedback . . . 134
11.2 Adaptive Document Recommendations . . . 135
11.3 Evaluation . . . 135

12 Future Work **137**

12.1 Adaptive User Needs . . . 137
12.2 The Synonym Problem . . . 138
12.3 Sources of Document Recommendations . . . 139

A Simulation Study **151**

A.1 Sample Simulation Run Record 151

B Evaluation Questionnaires **153**

B.1 Entry Questionnaire 153

B.2 Post Search Questionnaire 156

B.3 Exit Questionnaire 159

B.4 Comparison Questionnaire 160

C Profile Structure **161**

C.1 Sample XML Profile 161

C.2 Profile XML Definition (XSD) 165

List of Figures

4.1 The data model 35

5.1 Registration / Login interface 40

5.2 The search engine interface, illustrating a sample results page 41

5.3 Two examples of query-biased summaries of search results 42

5.4 The personalized home page, displaying additional documents discovered by the system . 43

5.5 Document bookmark in Colombus 44

5.6 The profile manager 44

6.1 Colombus profiling process overview 47

6.2 Profile hierarchy in Colombus 55

7.1 Relevance of topics in the second scenario 72

7.2 Relevance of topics in the second scenario 74

8.1 Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the first scenario 78

8.2 Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the first scenario 80

8.3 Correlation between the query terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the first scenario 80

8.4 Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the first scenario 81

8.5 Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the first scenario 82

8.6 Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the first scenario 82

8.7 Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=10, p=10, e=10) in the first scenario 83

8.8 Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=10, p=10, e=10) in the first scenario 83

8.9 Average recommendation precision of each term weighting scheme aggregated for all searcher stereotypes in the first scenario 85

8.10 Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario 87

8.11 Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario . . . 88

8.12 Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the second scenario 88

8.13 Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario 91

8.14 The profile evolution trend (PET) results of each term weighting scheme averaged for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario 92

8.15 The profile evolution trend (PET) results of each term weighting scheme averaged for searcher stereotype with tuple (t=5, p=30, e=30) in the second scenario 92

8.16 Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the second scenario 93

8.17 The profile evolution trend (PET) results of each term weighting scheme averaged for searcher stereotype with tuple (t=0, p=100, e=20) in the second scenario 93

8.18 Average recommendation precision of each term weighting scheme averaged for all searcher stereotypes in the second scenario 94

8.19 The profile evolution trend (PET) results of each term weighting scheme averaged for all searcher stereotypes in the second scenario 95

8.20 Average recommendation precision of a range of threshold settings averaged for all searcher stereotypes in the second scenario 96

8.21 Average recommendation precision of each term weighting scheme aggregated for all searcher stereotypes in the third scenario 98

8.22 The profile evolution trend (PET) results of each term weighting scheme averaged for all searcher stereotypes in the third scenario 99

8.23 Average recommendation precision of a range of threshold settings averaged for all searcher stereotypes in the third scenario 101

9.1 An example task briefing 109

9.2 The SearchIT search interface, illustrating a sample results page 111

10.1 Questionnaire data on how accurate user needs are reflected on their profile 116

10.2 Questionnaire data on how effectively user needs are captured in profiles 117

10.3 Explicit profile changes per evaluation day averaged for each experimental subject 118

10.4 Questionnaire data on explicit profile changes during the user study 118

10.5 Questionnaire data on the adaptivity of Colombus during the user study 119

10.6 Questionnaire data on the effectiveness of document recommendations during the user study 120

10.7 Questionnaire data on the interestingness of document recommendations 121

10.8 Questionnaire data on the novelty of document recommendations during the user study . 122

10.9 Recommendation access per evaluation day averaged for each experimental subject . . . 122

10.10 Searches issued per evaluation day averaged for each experimental subject 123

10.11 Questionnaire data on "How would you rate the systems performance in general" 124

10.12 Questionnaire data on "How much the following system helped you in your daily searching experience" 124

10.13 Questionnaire data on "How do you think the following system compares to your favorite search engine" 125

10.14 Questionnaire data on "With Colombus/SearchIT, I was able to find the documents I want" 126

10.15 Questionnaire data on "Which system did you prefer" 126

List of Tables

5.1 Interaction with the user interface 39

6.1 Example of term extraction table for the binary term weighting scheme 51

6.2 Example of term extraction table for TF weighting scheme 52

6.3 Example of term extraction table for TF-IDF weighting scheme 53

6.4 Statistics on storage and time requirements of the XML-based representation scheme . . . 60

8.1 Marginal difference in recommendation precision between search our tions for each term weighting scheme 79

8.2 Marginal differences in profile correlation across all search iterations for three simulated searchers for each term weighting scheme 84

8.3 Average performance of threshold settings and term weighting schemes on a per-iteration basis 89

8.4 Marginal profile evolution trend (PET) difference between each term weighting scheme and the baseline averaged for all searcher stereotypes in the third scenario 100

8.5 Marginal recommendation precision difference between each term threshold combination averaged for all searcher stereotypes in the third scenario 102

9.1 Inexperienced and experienced subject characteristics 108

9.2 The list of proposed topics for the user study 110

12.1 The evolution of an experimental task 138

Part I

Introduction

Chapter 1

Introduction

1.1 Introduction

Never before has access to the world wide web been available to so many people. The recent explosion of the internet along with the introduction of ubiquitous computing have increased the size of the Web by several orders of magnitude. Due to these technological advancements and the proliferation of computers, people generate tremendous amount of information. Web search engines, designed for discovering documents online, are very popular and generally perceived to do an excellent job in finding relevant information on the web. Given an accurate description of an information need in the form of query statements, they locate and present a set of relevant documents to searchers.

However, constructing query statements representative of searchers' information needs is typically a problematic activity (Ingwersen & Willett 1995). Often, the uncertainty of users as well as their lack of training on effective searching techniques make the query formulation process a challenging task. The problem is amplified after studies on search engines logs have denoted the short length of query statements, 2.35 terms on average, while two thirds of computer users issue queries of 2 terms or less (Jansen & Spink 2005, Jansen, Spink, Bateman & Saracevic 1998). Such short representations fail to describe information needs accurately. Also, recent studies have highlighted the shortcoming of web

search engines, focussing mainly on the lack of personalization features (Keenoy & Levene 2005). Further research on modern search behaviour showed that users interact only with a limited number of results, usually on the first page (Beitzel, Jensen, Chowdhury, Grossman & Frieder 2004). Uncertain about the availability of other relevant documents most people end their search sessions prematurely after one or two iterations. Furthermore, most of the time, they initiate new sessions regarding the same topics, for example things that relate to their work.

Personalized retrieval assistants are sought to facilitate information access and relieve users of the burden of query formulation and the information discovery process. Information filtering systems in the form of recommendation services (Zhu, Greiner & Haubl 2003, Balabanovic 1997), query expansion assistants (Harman 1988, White & Marchionini 2007) and collaborative agents (Resnick, Iacovou, Suchak, Bergstorm & Riedl 1994), to cater the needs of similar minded people, have gathered much attention over the years.

However, the role of personalization in retrieval systems has not been explored properly. Despite the fact that research has focused on improving existing ways of personalization, the drifting nature of user needs has received little attention. Long-term user interests gradually evolve after user interaction with new information. In addition, temporal topics may appear unexpectedly. Given the content generation and mature technological scenario, new classes of personalized and context sensitive access patterns are possible. For example, personal recommendation assistants can pro-actively profile searchers and adapt to their changing interests. The development of such class of techniques and systems is hampered by the lack of proper evaluation methodologies and collections.

There are two components to information filtering systems: a user modelling component as well as a recommendation and presentation subsystem. The majority of adaptive information retrieval tools employ relevance feedback (RF) gathering mechanisms to capture the user intentions and create their *profile* (Belkin & Croft 1992), an appropriate snapshot of their information needs. RF techniques commonly rely on explicit ratings of the objects in the information domain. These ratings are then processed to calculate user preferences. Alternatively, implicit feedback gathering techniques (Kelly & Teevan 2003, Nichols 1997) infer document relevance by observing user interactions with the retrieval systems.

Document reading time (Kelly & Belkin 2001, Parsons, Ralph & Gallagher 2004) and click-through data (Joachims 2002, Xue, Zeng, Chen, Yu, Ma, Xi & Fxan 2004) are often recorded during the relevance feedback process.

This thesis presents the development of an adaptive recommendation assistant based on implicit knowledge to profile user needs and provide personalized recommendations. A number of implicit term selection models have been developed and deployed as part of the user profiling process. Also, I investigate the nature of concept drifts in a hierarchical multi-facet profile environment and propose novel methods to create an adaptive profile learning scheme for personalized information filtering systems.

Evaluation of such systems is a delicate issue. Profile learning schemes can be considerably different, ranging from statistical-based techniques to complex machine-learning algorithms. Due to the lack of an established evaluation methodology and the vast differences in the design and implementation of adaptive information filtering tools, comparison of such systems is not always feasible. Current methodologies in information retrieval evaluation favour offline lab experiments, which exclude the user from the evaluation phase (Harman 1992*a*). While user studies give an indication of a system's performance in realistic conditions, the need to field a fully engineered system and build up a community of users has led research towards lab-based experiments. Simulation-based evaluation methodologies can be employed to benchmark a number of system parameters to optimize the performance of profile-aware information assistants.

We have established an evaluation framework, based on a hybrid approach, by combining large-scale searcher simulation studies with online, user-centred experiments. Simulation experiments are augmented through realistic searcher representations by taking into account user characteristics, like patience and experience. Simulations are executed in large batches to discover optimal settings of system parameters. On the other hand, user-centered experiments benchmark subjective aspects of the system, such as the level of adaptivity and the overall satisfaction. Along this direction, a basic search system is employed, as a baseline, to allow comparisons between *actively adaptive* search recommenders and web search engines.

1.2 Thesis Layout

This thesis is split into the following chapters:

- Chapter 2 provides some required background knowledge on information retrieval and recommendation systems and presents, in detail, the main reasons underlying the development of this system, as well as a review of past solutions in this area.
- Chapter 3 focuses on the main concepts behind recommendation services, aiming to model evolving user needs and discusses feedback gathering techniques, both implicit and explicit, widely used in search assistants, and outlines the implemented algorithms used in this aspect. Further on, it covers concept drifts in the context of information retrieval, while an attempt is made to categorise and explain the different types of concept changes.
- Chapter 5 analyzes the system interface features facilitating information discovery, recommended document access and profile management.
- Similarly, chapter 4.2 focuses on the representation of user profiles and provides a general overview of the persistence manager implemented in our system.
- Chapter 6 describes, in detail, the system architecture and explains the various algorithms used in our system, such as the term extraction technique and the profile generation algorithm.
- Chapter 7 provides a detailed analysis of the experimentation methodology employed during the simulation-based evaluation, while chapter 10 presents the results along this direction.
- Chapter 9 presents the experimental methodology employed for the user-based evaluation of Colombus, while the results on this aspect are analysed along chapter 8.
- Last but not least, chapter 11 presents the main points that this thesis has made, and also highlights some issues for future work.

Chapter 2

Background and Motivation

2.1 Relevance Feedback

With the growth of the World Wide Web the need for tools to address problems with information overload has become more apparent (Nelson 1994). However, in many situations the information seeking experience is less than satisfactory: often searchers have difficulty expressing their search needs and, thus, finding relevant information. The main reason for this is the lack of effective search interaction and retrieval tools. The existing tools are often ineffective for all but the most simple search tasks (Dennis, McArthur & Bruza 1998). User interaction with the search engine interface is a three-stage process: Query formulation, relevance assessment and query modification (Ingwersen & Willett 1995). To build an effective search tool one has to address the problems imposed during query formulation and relevance assessment.

Relevance feedback (RF) techniques attempt to improve the representation of information needs by allowing searchers to directly express their notion of relevance with respect to individual documents. RF has been employed in several classes of personalization systems. Driven by the need for better representation of information needs, RF was initially introduced to support basic query expansion (QE) (Rocchio 1971). However, its success in inferring the user's notion of relevance on a per-document basis

has lead to a subsequent adoption by information filtering and recommendation systems.

Relevance feedback approaches are based on a feedback gathering scheme, either *explicit* or *implicit*. In the former, object ratings of predefined scale are provided explicitly by users, while implicit feedback gathering techniques infer object relevance in a transparent fashion, by monitoring searcher interaction with user interface components. Explicit RF techniques have dominated past scientific literature (Joachims, Freitag & Mitchell 1995, Armstrong, Freitag, Joachims & Mitchell 1995, Pazzani, Muramatsu & Billsus 1996).

2.1.1 Explicit Ratings

In our everyday lives, we constantly rate things explicitly: from teacher grades to performance evaluation reports, all employ some form of rating system to assess the performance of an object (Nichols 1997). In a similar fashion, several online systems have adopted such an approach. For instance, MovieLens (Miller, Albert, Lam, Konstan & Riedl 2003) offer movie recommendation services to their members by creating a user profile based on their subjective rating of films. Grouplens (Resnick et al. 1994), the predecessor of MovieLens, follows a similar technique to implement the collaborative filtering of Internet news. Grouplens users rate articles after having read them and the system aggregates ratings and analyses for future use. Since both these systems originate directly from user explicit judgments, they lead to an accurate estimation of information requirements.

Instead of using a collaborative approach, Kumaran (Kumaran & Allan 2006) has proposed the use of lightweight explicit feedback gathering techniques for correcting false or unstructured information. They have designed interaction strategies to handle spelling mistakes, punctuation recognition as well as identifying patterns in top-ranked documents. However, even less noticeable explicit feedback approaches are considered expensive and the results may not become immediately apparent .

Typical search behavior is characterized by the act of document reading. Explicit relevance ratings change the normal pattern of searching by asking users to examine and assess documents in the result list (Nichols 1997). Such additional activities impose a great burden and confuse searchers. Also, explicit RF techniques disregard user knowledge on the current topic. Users are often unclear about their search interests. They browse for more information to clarify their need and re-formulate their query accordingly.

The uncertainty in their search episodes increases the cognitive load during explicit RF, as users must decide on the relevance of a document possibly with a lack of confidence.

Several studies have examined the effectiveness of typical document *surrogates* in modern web search engines (Joho & Jose 2008). A surrogate refers to the representation of a document in the result list. Common document surrogates used in search engines contain a title, snippet, URL and possibly the file type in the search engine. Joho (Joho & Jose 2008) has denoted that an inadequate representation of information objects can mislead relevance assessments. In effect, explicit relevance judgments are directly influenced, since the simplicity of modern search engine surrogates cannot adequately describe the relevance of a document with respect to the user's query.

Finally, the use of explicit ratings imposes privacy issues that have to be resolved (Keenoy & Levene 2005). Irrespective of the underlying reason, users are not always comfortable in providing direct indications of their search interests. Due to the obtrusive nature of explicit ratings, not many searchers are willing to provide them. Hence, the performance of profile capturing and recommendation algorithms of such systems degrades, due to the dearth of ratings. In social filtering systems based on explicit feedback gathering policies, the sparsity of RF judgments can often render such systems unusable, since there are few previous assessments to learn from.

2.1.2 Implicit Ratings

Implicit relevance feedback gathering techniques have matured to an unobtrusive, equal alternative to explicit ratings (Keenoy & Levene 2005, White, Ruthven & Jose 2002*b*). Such techniques passively monitor user interactions with the system in order to estimate their search interests (Morita & Shinoda 1994). Click-throughs, time spent viewing a document and mouse gestures are among the possible sources of implicit feedback (Kelly & Teevan 2003). The main benefit of implicit feedback, over explicit ratings, is that they remove the cognitive cost of providing relevance judgments explicitly. Although the accuracy of implicit approaches has been questioned (Nichols 1997), recent studies have shown that they can be an effective substitute for explicit relevance feedback (White et al. 2002*b*). Since implicit judgments are derived transparently, they contain less indicative value than explicit ratings, but can be gathered in

large quantities and aggregated to calculate document relevance.

2.1.2.1 Sources of Implicit Ratings

Implicit feedback techniques observe searcher interactions with the user interface to understand and estimate user preferences and interests. There are several types of feedback that can be implicitly captured. Typically, the quality of implicit evidence and document relevance is directly related to the source of feedback. Past scientific literature has focused on classifying implicit feedback sources based on the underlying user behaviour (Kelly & Teevan 2003, Stevens 1993, Nichols 1997).

InfoScope (Stevens 1993), a system for filtering Internet discussion groups modeled users by using three sources of implicit evidence: whether a message was read or ignored, whether it was saved or deleted, and whether or not a follow up message was posted. Monitoring the reading time of a document could also be used as an implicit feedback source. Morita & Shinoda (Morita & Shinoda 1994) concluded that the time spent reading documents on the web is closely related to the degree it suits the needs of each user. An alternative measure of implicit feedback is to assume that all printed documents are relevant and therefore try to detect the user's prole from this kind of behavior. Kim has adopted a similar approach in (Kim, Oard & Romanik 2000).

Nichols (Nichols 1997) builds on the work of Infoscope (Morita & Shinoda 1994, Nichols, Twindale & Paice 1997, Stevens 1993) to categorize actions that can be observed during user information seeking episodes. He indicates that although limited evidence shows a great potential in implicit ratings, their effectiveness remains unproven by formal evaluation means. In a similar fashion, Oard and Kim (Oard & Kim 2001) provide a taxonomy of implicit feedback types into four main categories, based on the purpose of the observed behavior. The four categories are: "*Examine*" (e.g view), "*Retain*" (e.g. print), "*Reference*" (e.g. reply) and "*Annotate*" (e.g. publish). Kelly (Kelly & Teevan 2003) extends this taxonomy of observable behaviors by adding a "*Create*" category to represent actions related to the creation of new information. Their work has also focused on classifying existing scientific literature on implicit feedback to the category of behaviors they exhibit.

Despite the fact that implicit feedback has been categorized into four categories, the behavior of searchers

in online information filtering systems can be described using actions from the "*Examine*" and "*Retain*" categories. User behaviors that fall in the "*Reference*", "*Annotate*" and "*Create*" require control over individual web services and applications and, thus, cannot be captured. The relevance feedback policy in Colombus combines actions from both the "*Examine*" and "*Retain*" categories. In this aspect, we monitor document examination, usually referred to as *click-through*, summary viewing and a bespoke form of bookmarking to estimate user preferences.

Click-through actions reside into the "*Examine*" category. Such data can be easily captured in real-time at no considerable computational cost. Also, due to the abundance of click-through data in search engine query logs, they facilitate large-scale evaluation experiments of information filtering tools (Oard & Kim 2001). Several applications have been proposed for feature extraction (Belkin 2000, Chien-Kang, Lee-Feng & Yen-Jen 2003) and query expansion (Cui, Wen, Nie & Ma 2003). However, their quality has been questioned, while their benefit for estimating user interests is still unclear.

Web search behavior has been characterized by the act of document browsing and reading. In a typical scenario, users issue queries and browse through the results for relevant documents. Due to the incompleteness of document surrogates and the ambiguity of information needs, searchers often navigate through irrelevant information until they define and clarify their search task. Arguably, such behavior can result in noisy click-through data, and, in effect, relevance judgments. Also, the limited interaction of users with the search interface often contributes to the sparsity of click-through data.

Although document examination does not always dictate document relevance, it provides an indication of user intentions and search goals. Xue (Xue et al. 2004) discovered that 82% of queries are related to subsequent click-through actions. Similarly, Joachims (Joachims 2002) propose a method of incorporating click-through data in a retrieval function for personalized searching. His method takes into account the relative position of clicks in a result list as training data. The experimental results show a clear improvement in web search experience from the use of click-through data.

Although the effectiveness of implicit relevance feedback and click-through data has been extensively studied, there are only a few fully engineered systems that employ this form of RF for document recommendations (Xue et al. 2004). In this thesis, I explore the performance of profile learning approaches

based solely on implicit feedback techniques and define a set of user actions with the user interface around which I build a number of implicit feedback gathering models for adaptive information filtering. Section 6.4 covers the implicit feedback gathering policy and process in greater detail.

2.2 Personalization System

During the *query formulation* process, information needs are expressed in the form of query terms. In this sense, the retrieval effectiveness of information retrieval tools is directly defined by user queries. Given the right set of query terms, search engines strive to find relevant document with respect to user needs. However, due to the uncertainty of users and the cognitive load in defining their search requirements, the process of query formulation can be problematic.

Also, past literature on this area indicates that typical web search behavior is characterized by short queries and limited interaction with search results (Silverstein, Henzinger, Marais & Moricz 1998). In fact, 83% of searchers examine documents only within the first two result pages (Jansen & Spink 2005), while 66% of user queries consist of a maximum of two terms. Unaware of the presence of other relevant pages, users often cease their search iterations prematurely. Also, due to the ambiguity of concise queries, information needs are often unclear. For instance, the query "*JAVA*" can be in the context of programming languages, Indonesia or coffee beans. Interactive query clarification techniques and long-term profiling of user needs are among the ways to address these problems.

Recently, commercial search engines have taken a step closer to personalization, mainly on the user interface level. By offering *customization* services, such as personalized access to news, weather and stock prices, modern search engines have attempted to address the problems mentioned earlier. In this aspect, Yahoo offers *My Yahoo!* (Manber, Patel & Robison 2000), while Google offers *iGoogle*, both based on explicit means from user preferences. However, searchers have refrained from using such primitive features (White, Jose, Van Rijsbergen & Ruthven 2004). On the other hand, typical web search methodology has remained relatively consistent over the past few decades, while proper personalization features have yet to become available in commercial search engines (Khopkar, Spink, Giles, Shah & Debnath 2003).

Novel personalization systems have emerged to bridge the uncertainty gap in web search which is further amplified by short searching times and few query words.. There are many approaches to providing personalization services for web search environments. Query expansion (QE), recommendation systems and result re-ranking are well-established techniques towards this aspect. Iterative personalization systems (as opposed to short-term per-session learning approaches) adopt RF-based approaches for effectively learning the user's profile. The remainder of this section focuses on providing additional information on the characteristics of several types of personalization in the field of information retrieval.

2.2.1 Personalization through Recommendations

Keeping up-to-date with their regular long-term information needs requires time and effort from searchers. With the rapid growth of online information sources, there is an increasing interest for tools to support and facilitate web browsing. Personalized recommendation systems serve the purpose of assisting users in their search experience and accommodating their information needs. By recording data from past search episodes, such systems strive to discover new relevant information with respect to user interests.

In essence, the recommendation process is closely associated with information seeking. Uncertain about their real information needs, users often search for more information in order to define and clarify their search interests. In this sense, a recommendation system can be seen as a support mechanism for users to articulate their search needs. The recent surge in research activity on information assistants has resulted in a range of novel profile learning approaches for recommendation.

Letizia (Lieberman 1995) is a web agent that silently follows the user in his search quests and provides on-demand recommendations. When the user navigates to a web page, the agent attempts to anticipate items of interest by doing concurrent, autonomous exploration of links from the current page. Recommendations are calculated based on the similarity of the contents of each page compared to the single-facet user profile.

In Syskill & Webert (Pazzani et al. 1996), interests are stored as separate topic-specific keyword vectors in the user profile, which is consulted and analyzed to collect novel information. Profile learning is realized through a machine-learning approaches, which record user feedback in the form of explicit judgments

on 3-point scale (e.g. hot, lukewarm, cold). Navigation and searching through Syskill & Webert are coordinated through topic-dependent index pages which contain links to other information providers.

In WebWatcher (Armstrong et al. 1995), searchers indicate their search goals, via explicit means, in the form of keyword descriptions. The system suggests appealing web pages by looking at the structure of hypertext. However, personalization features were integrated on a community-basis, like in information sharing systems, without taking into account search needs of individuals. Mladenic et al have identified the shortcomings of the original WebWatcher and designed a system tailored to individual user interests. Personal WebWatcher (Mladenic 1996) applies a Naive Bayesian classifier on frequency vectors to generate a model of user interests. Click-through data are recorded and further processed to extract indicative features from relevant documents.

News filtering and personalized news portals have arisen as another flavour of recommendation systems (Chen & Sycara 1998, Kamba & Bharat 1995). For instance, Webmate (Chen & Sycara 1998) accompanies searchers during their information seeking tasks and generates a personalized newspaper of daily news. Explicit relevance judgments are accommodated during the profile learning process, while interests are represented as separate keyword vectors in the user's profile. The *Krakatoa Chronicle* (Kamba & Bharat 1995) is a personalized newspaper which monitors implicit user interactions with the system to generate a vector-based profile. It differs from other online news filtering systems by focussing on advanced UI aspects, such as dynamic layout control. Also, an approach to collaborative feedback has been implemented in the form of community scores for articles.

To our best knowledge, only few of the above systems address issues with information overload and profile learning. Section 2.1.1 has illustrated the problems with explicit relevance feedback. In fact, due to the cognitive load of explicit ratings, the benefits from facilitating information access through document recommendations are not always become apparent. Ideally a recommendation system would recognize and adapts to drifting interests with no extra effort from the user. By gathering relevance feedback through implicit means, Colombus has followed an effortless approach to document recommendations.

2.2.2 Personalization through Query Expansion

Query expansion systems assist users during the query formulation process, by augmenting their original query in an interactive (Ruthven 2003) or automatic fashion (Carpineto, De Mori & Romano 1998). While automatic QE techniques update and re-submit the original query transparently to the user, some systems have adopted a more interactive approach. Interactive query expansion provides the user more control of the expansion process by allowing searchers to choose from a list of expansion terms for inclusion in the augmented query. The scope of information gathered during the current search session has created another conceptual separation. Rather than expanding queries based on a profile created across all search sessions, several QE techniques work on the course of a single search iteration. The later approach is a form of short-term query assistance.

Query expansion techniques have been found to improve the retrieval effectiveness of the underlying system, especially for the first few iterations (Joachims 1997, Harman 1992*b*). While Rocchio's term weighting formula (Rocchio 1971) has been extensively used in this area, similar vector-space (Baeza-Yates & Ribeiro-Neto 1999) and probabilistic based approaches have also been employed in this aspect.

Carpineto (Carpineto et al. 1998) has examined the concept of relative entropy in the context of automatic query expansion. He developed a weighting formula based on the Kullback-Liebler distance (Kullback & Leibler 1951) to derive candidate terms for query expansion. Based on a set of semantic factors, like the distribution of terms across relevant documents and across the whole collection, his formula discriminates good expansion terms from poor ones. Similarly, other researchers (Harman 1992*b*, Ruthven 2003) experimented with query expansion through probabilistic term weighting. Term weighting based on the WPQ (White et al. 2004) method of ranking terms was used to derive a set of expansion terms from relevant documents. Ruthven et al (Ruthven 2003) investigated the effect of optimal decisions in interactive query expansion over automatic strategies during lab-based experiments. It was demonstrated that optimal QE is difficult to achieve through interactive means, due to user interface restrictions.

2.2.3 Personalization through Result Re-ranking

Personalization features have yet to appear in ranking algorithms employed by commercial web search engines (Keenoy & Levene 2005). Their ranking approach disregards individual user needs; the same query returns the same ranking of results. However, the lack of profiling features combined with the short length of user queries leads to the ambiguity of information needs in web search engines. During the past decade, a growing trend towards personalized ranking approaches have been observed, motivated by the need for customized search results. Such techniques are similar to query expansion through long-term user profiling, as opposed to short-term query expansion during a single search session. In effect, *result re-ranking* approaches incorporate personalization features directly in the document ranking function to increase the ranking of documents based on the user preferences.

Personalized PageRank (PRR) (Jeh & Widom 2003) extends the original PageRank algorithm (Brin & Page 1998), to incorporate personalization features directly in the ranking function. The profile of each searcher is created and represented as a set of page preferences, captured through explicit means. In a similar fashion, topic-sensitive PageRank (Haveliwala 2003) returns a topic-oriented page ranking by polling the Open Directory Project (ODP).

Last but not least, PROS (Chirita, Olmedilla & Nejdl 2004) is a personalized ranking platform for web search environments that builds on the work of PRR. Instead of asking users to create their profile explicitly, PROS monitors search history and reading time, through a web proxy, and captures bookmark actions, to gather preferences via implicit means. Every time a query is issued, the system discovers pages related to the user's preferences and updates the result ranking. The experimental results presented illustrate a significant improvement over the baseline system.

2.3 Evaluating Personalization Systems

The evaluation of indexing and retrieval algorithms for typical IR tools, such as search engines, follows a well-defined methodology. Initiatives, such as the Text Retrieval Conference (TREC), create test collections consisting of a large number of documents and groups of queries or topics of interest. The

relevance of documents with respect to each particular query is manually assigned by human assessors. Driven by the need for a common evaluation methodology of IR systems, the TREC initiative assesses systems in the grounds of precision and recall measures (Hull 1997). However, it fails to address the evaluation of personalized retrieval tools (Borlund 2000, Jansen & Pooch 2001).

Evaluating personalized systems and search assistants is a challenging task due to the variety of possible scenarios that can occur. Variations in recommendation algorithms and system implementations make evaluation even harder. As Konstan and Riedl (Konstan & Riedl 1999) indicate, existing approaches to evaluating recommender systems can be divided into two categories. In off-line evaluation techniques, the performance of a recommendation mechanism is evaluated on existing datasets, while in on-line evaluation methodologies, the performance is evaluated on users of a running recommender system.

Standard, offline evaluation methodologies (both collection based and interactive), applied to IR systems, are not applicable to personalization systems. First of all, they exclude personalization and user profiling features from the evaluation stage (Harman 1992*a*), but these constitute the very grounds for the motivation of these systems. Second, they assume queries arrive in batches, ignoring differences in characteristics and search habits of individuals, such as patience and fatigue. Similarly, interactive evaluation methodologies, largely laboratory-based, do not allow the evaluation of systems dealing with long-term information needs and drifting concepts.

From a TREC perspective, the filtering task (Hull 1997) attempts to simulate an information filtering application by assessing an incoming stream of objects. Filtering task aims to improve the routing task and set the grounds for a more realistic simulation environment for information filtering systems. In particular, the documents in the collection have been assigned a date stamp and are processed by the underlying system in date order. Participating systems are assessed on the grounds of utility, which represent the cost of each document, as well as average set precision, which is defined as the product of precision and recall values. TREC filtering task has been criticised for providing too much training data, which is unrealistic in a real world environment. The *adaptive filtering subtrack* follows a similar approach to the filtering task, but participating systems start with only a single topic description, while no previous training has taken place. Despite these enhancements, there is still a considerable gap between

the TREC filtering experiments and operational systems with respect to the lack of user interaction during the evaluation process and the ability to measure the adaptivity of filtering systems (Hull 1997).

Often, the performance of profiling algorithms and systems is benchmarked against TREC datasets by employing well-established evaluation measures on user profiles and document recommendations (Lanquillon & Renz 1999, Klinkenberg 2004) or query expansion. Precision and recall over recommendations, as well as accuracy of user profiles are commonly adopted for the evaluation of recommendation assistants (White, Ruthven, Jose & Van Rijsbergen 2005). However, evaluation methodologies for adaptive context-aware systems require more profile-oriented measures than basic precision and recall. As part of the evaluation methodology I propose a novel technique to capture the trend of user interests during the profiling process.

The topic Detection and Tracking (TDT) research community investigates evaluation methods for news categorization systems, by their ability to cluster similar stories together (Allan, Carbonell, Doddington, Yamron & Yang 1998). The TDT evaluation program consists of various subtasks: (i) *segmentation* of stream to news stories, (ii) *new event detection* for finding unseen information, (iii) *clustering* of stories to semantically-similar groups, (iv) partially supervised tracking of incoming news and (v) *story link detection* for deciding whether two stories are on the same topic. The performance of participating systems is measured through a novel cost function, based on the system's miss rate. However, TDT evaluation does not include user interface issues or the complexity of search tasks (Allan, Harding, Fisher, Bolivar, Guzman-Lara & Amstutz 2005). Also, no measure to evaluate the adaptivity of participating systems is provided.

On the other hand, online user-centred evaluation studies can be employed to accurately measure the effectiveness of system algorithms and the usability of user interfaces. However, such techniques can only be carried out based on a specific system configuration and, therefore, fail to benchmark the performance of different parameter settings. Also, Kostant et al has highlighted the problems of such approaches (Konstan & Riedl 1999). The need to field a fully engineered system and build up a community of users eager to use a prototype system frequently for long-term studies is the major drawback of user-centred evaluation techniques. But as long as the system is in a working condition and the participants have

been found, on-line evaluation techniques are certainly more accurate than lab-based alternatives.

Simulated user evaluations are an alternative but need to be developed from a methodological perspective. Such techniques simulate the presence of searchers, during the evaluation phase, by mimicking a typical web search process. In effect, a *simulated searcher* is an artificial user created with parameters and properties that are as close as possible to a real world web search user. The benefit of evaluation experiments based on searcher simulations is that they can be carried out in large volumes, so the experimental results are more realistic than other offline strategies. In addition, by simulating different user models, the performance of systems under different contexts and behaviours can be measured.

Although simulation-based methods have been used to test query modification techniques (Harman 1988, Ruthven 2003) and to detect concept shifts (Lam, Mukhopadhyay, Mostafa & Palakal 1996, Mostafa, Mukhopadhyay & Palakal 2003), to our best knowledge not much research has been carried out in creating realistic searcher models for evaluating information filtering systems. White et al (White et al. 2005) created searcher simulations for evaluating implicit feedback models. The simulation assumes the role of a searcher, browsing the results of an initial retrieval. The information content of the top-ranked documents in the first retrieved document set was the space explored while simulating. In the simulation searchers were modeled using a number of different strategies depending on their interaction with relevant/non-relevant information. Their research tries to model only certain phases of the search process like clicking the results and to some extent the process of looking and identifying the results to click, while it does not consider searcher characteristics (e.g. patience, fatigue).

In this study, I present a framework for evaluating recommendation systems based on hybrid approach between searcher simulation techniques and user-centered experiments. The methodology consists of two stages: (i) *system tuning* and (ii) *user evaluation*. First, a number of different user models are instantiated to benchmark and optimize the performance of different parameters. The effectiveness of profile learning and document recommendation is assessed against typical evaluation measures, such as profile accuracy and precision as well as a bespoke profile evolution trend measure (PET). Chapter 7 presents the simulation stage in detail. At the second stage, I carry out a task-oriented, user-centered evaluation methodology with 15 experimental subjects who agreed to use the system for 15-20 days. The

results of the user experiments are presented in more detail in chapter 8.

Part II

User Profiling

Chapter 3

User Profiling

3.1 Dynamics of Search Needs

In order to integrate personalization features in IR tools we must first understand the nature of search needs. There is ongoing research in the user profiling community in order to determine and specify user requirements from search tools. Search *"episodes"* are defined as the number of search iterations from the appearance of a new user interest until the information need has been satisfied.

In this section, I present the dynamics of search needs, ranging from long-term episodes, to short, abrupt information requirements. Also, I focus on the implications and marginal cases which profiling schemes must take into account when monitoring user search iterations.

3.1.1 Long-term vs. temporal search needs

An information need can be triggered from events such as changes in the user's environment, job responsibilities or hobbies. Long search episodes dealing with the same topic are usually initiated by our work environment or persistent information interests. In contrast to such long-term needs, a sudden change in the user's attention can create temporal interests. For instance, when someone wants to buy a car, he

will change his usual search agenda to focus on finding a suitable vehicle. When he finally finds what he wants, it's likely that this information need is satisfied and the search for a vehicle will stop there. To complicate things even further, temporal interests can persist in the user search behavior and eventually become long-term. A possible example is when someone decides to redecorate his house, but as he finds more information online he gets increasingly interested in gardening, which ultimately evolves to personal interest or hobby.

This indistinct nature of user interests affects the design of profiling algorithms for personalized filtering systems. Several researchers in the past have clearly differentiated between long and short-term user interests by identifying and accommodating them differently. For instance, Bilsus and Pazzani (Pazzani et al. 1996) have created a news recommendation system based on a combination of explicit and implicit relevance feedback to infer user preference. The profiling component distinguishes between long and short term user needs by monitoring feature patterns. Long term needs are accommodated through a naive Bayesian classifier that monitors user interaction for high frequency features. On the other hand, shorter term interests are taken care off by activating a K-Nearest neighbour approach combined with TF-IDF term weights.

In contrast with these approaches, another part of the user profiling community argues that user interests, both long-term and temporal, should be catered to in a common way, biased by the fact that any concise information requirement can eventually persist in the user's search interactions and become long-term. For instance, Carreira (Carreira, Crato, Gonçalves & Jorge 2004) has created WebClipping, a system responsible for retrieving news from the internet combined with a palmtop browser for accessing news on the go. In terms of the relevance feedback policy, a combination of four implicit feedback metrics are employed, while the profiling component of the system handles both short and long term interests in the same way by triggering a naive Bayesian classifier to select documents.

A similar approach has been employed in Colombus, as both temporal and persisting interests are accommodated in the same way through a K-nearest neighbour algorithm that uses a pair of thresholds to classify and categorize incoming information (terms and documents) into a hierarchical user profile.

3.2 Detecting and Adapting to Drifting Concepts

Further analysis of user search needs have signified their volatile nature. Information requirements change due to users collecting more information on the topic or changes in their environment. Such changes can be expressed as slight drifts in the user's perspective on a subject, or abrupt shifts. In any case, the user's perception of relevance changes in such a way that past search iterations no longer accurately represent his current information needs.

The need to incorporate concept drift detection mechanisms in our adaptive retrieval models is crucial. Past research on this area has shown that the performance of static profiling schemes, with no adaptation capabilities, degrades continuously. Retrieval mechanisms that profile user needs in order to provide a form of personalized service need to rapidly identify such concept changes and adapt effectively.

Drifts in the users' perspective do not occur solely in user profiling systems, but are noticed in our everyday lives. A decision change, like cancelling a meeting, is a simple example of a concept change scenario. In the context of information retrieval, concept drifts occur at a much higher rate. In general, interaction with new information or new states of things is likely to trigger a change in concept (Belkin 1997). During a typical web browsing task, as users collect new information on particular areas, their queries and interaction with the search engine become much more focused. For instance, students will refine their searches as they attend more courses in the university and become more knowledgeable on the field. An ideal user profiling system must detect concept drifts and handle them effectively, but without imposing any cognitive load on the user. To complicate things even further, concept drifts range from slow, gradual changes to sudden, abrupt shifts.

The remainder of this chapter is arranged as follows: In subsection 3.2.1, I present the different types of concepts drifts and the implications in the underlying learning algorithms. A range of indications that signal a change in the user's point of relevance are identified and presented along section 3.2.2. Finally, a series of strategies to handle drifting concepts in profile learning environments are detailed in section 3.2.3.

3.2.1 Types of Topic Drifts

Several studies have established the fact that user interests not only aren't stable, but can change in many different ways (Psarras & Jose 2006). *Topic or concept drifts* are defined as slight changes in the point of relevance, while sudden, abrupt changes are often called *topic or concept shifts*. In fact, the nature and type of change is directly related to the cause that triggered such an event.

Global, world-triggered events often cause abrupt shifts in the user's point of relevance. For instance, a terrorist attack is likely to cause a temporal shift in the users' usual searching habits. However, there are cases where global events trigger gradual change of topic. For example, a researcher is likely to follow the latest innovations in his field of interest and adapt his search strategy and generally his research to take advantage of newer information.

In a similar fashion, gradual changes in the notion of relevance often occur as the user collects more information about a particular subject. In such cases, user queries become increasingly focused on specific aspects of the subject.

The theory of topic drifting is closely related to user search behaviour and the nature of user needs. An abrupt shift in the relevance perspective introduces a new topic of interest in the user's profile. Although this change of relevance seems to be temporal and short-lived, users can eventually get more interested in this area and extend their searches indefinitely. Therefore, the lifetime of such user interests is prolonged such that it is no longer temporal, but long-term.

3.2.2 Detecting Topic Drifts

Before handling topic drifts by employing adaptation mechanisms, profiling systems must initially detect them. Being able to distinguish between real changes in the point of relevance and noise is the main challenge of adaptive learning techniques.

A range of different indicators of concept changes have been proposed over the years (Lam et al. 1996, Klinkenberg & Renz 1998, Widmer & Kubat 1996). The FLORA framework attempted to adapt to the extent of the concept drift by monitoring the predictive accuracy and syntactic properties of the

evolving hypothesis. In a similar fashion, SIFTER (Lam et al. 1996) tracks the revision of user interests by observing drifts in the sequence of relevance feedback judgments. A Bayesian approach is employed to compute the probability that a change in concept has occurred. Klinkenberg (Klinkenberg & Renz 1998) categorized a set of measures to detect changes into three groups:

- performance measures (e.g. accuracy, precision, recall)
- classification model properties (e.g. rule complexity)
- data properties (e.g. attribute distribution)

They employed well-established evaluation measures, precision and recall, together with accuracy of the classifier to capture changes in user needs.

However, the application of such techniques largely depends on the implementation of the underlying learning strategy. For instance, monitoring the learning accuracy can be applied only to classifier-based learning approaches.

3.2.3 Handling Topic Drift

Systems and algorithms for handling drifting concepts are often equipped with filtering and weighting mechanisms, driven by the assumption that recent observations describe the searchers' need more accurately. The following sections illustrate the three main strategies for handling concept drifts.

3.2.3.1 Time window

The problem of learning concept drifts can be approached as the problem of finding a certain time point t where the topic has lost focus (Klinkenberg 2001). Such behavior can be temporal, in which case the topic will re-attract the user's interest, or permanent. An adaptive learning algorithm needs to identify point t and take into account samples gathered since then. In essence, the time interval between t and present can be apprehended as a window of data to learn from. The *time window* approach to adaptive retrieval closely follows this methodology.

Systems that employ a time window approach to learning data address concept drifts by selecting a number of examples from past search episodes. The range of examples to learn from is directly associated with the size of the window. Observations outside this window, i.e. older than a certain age, are instantly forgotten. Discovering the most effective window size, or in other words identifying point *t,* has always been the downside of this approach. In fact, the size of the window is closely related to the rate of concept drift. A narrow window indicates the volatile nature of a topic, while a wide one implies a much more stable user interest.

Klinkenberg (Klinkenberg & Renz 1998, Klinkenberg 2004) illustrated the deficiencies of fixed time windows. Such windows are based on the assumption that user needs change at a permanent rate. Instead of setting the time window to a fixed size, more context-aware versions of this approach automatically adjust the window size to the extent of the concept drift. Depending on the rate of change, the size of the window adapts to recognize and capture the change in the point of relevance. As the search need becomes more stable, the window size is increased, when a concept drift is suspected, the window size is reduces leaving old context behind. Past research on this area has shown that adaptive window techniques depending on complicated heuristics also result in performance decrease, due to the large parameter space (Klinkenberg 2004).

There has been an increasing amount of research on the context of time windows recently. Widmer (Widmer & Kubat 1996) developed a family of algorithms, the FLORA framework, to handle drifting concepts and adapt to user interests. Their system incorporates a three-descriptor vector to classify positive and negative data, as well as a reserve for examples that might become relevant in future searches. An adaptive time window is employed, in order to adapt to the extent of the current drift. Apart from the baseline algorithm (FLORA 2), two extensions were introduced to handle recurring concepts (FLORA 3), as well as noisy learning data (FLORA 4). FLORA systems have identified the abruptness of time window techniques in that older examples are completely forgotten and have introduced a partial memory to keep track of previous contexts.

Klinkenberg (Klinkenberg & Renz 1998, Klinkenberg 2004) developed an adaptive time window technique, where the size of the window was adjusted by monitoring three performance measures (accuracy, precision

and recall) as indicators of concept drifts. The presence of a concept drift is suspected by determining whether these measures are below a certain threshold value. This technique also differentiates between abrupt shifts and graceful drifts by performing an additional threshold-based test.

The downside of this technique is that it abruptly forgets older observations (Koychev 2001). However, searchers' history contains the evolution of their information needs and should not be excluded. Also, searcher interests are recurring; they may decide to return back to the shape of their initial information needs as happens for example when a dead-end is reached. A more robust method that gradually forgets past observations is presented in the next section.

3.2.3.2 Ensemble Classifiers

Several studies, such as (Bauer & Kohavi 1999, Dietterich 2000), suggest that a group of classifiers perform better than a single one. Driven by this assumption, ensemble learning maintains a group of classifiers over the topic descriptors and combines their judgments. In the context of topic change, base learners detect drifting concepts and adapt based on their votes on the learning data. Concept Primitive versions of this approach introduced a fixed number of *"experts"* to cover the topic features, but due to such a restrictive nature they could only be applied to problems where the size of the ensemble could be determined in advance. Dynamic ensemble classifiers in the area of drifting concepts have not received much attention till recently (Stanley 2001, Klinkenberg & Ruping 2002, Kolter & Maloof 2003).

The Streaming Ensemble Algorithm (SEA) (Nick Street & YongSeog 2001) creates new classifiers based on a fixed amount of data to improve the performance of the ensemble. New classifiers can either be added to the ensemble or replace an existing one with poor performance. (Kolter & Maloof 2003) have highlighted the shortcomings of this approach. They propose an ensemble of classifiers of different age (i.e. weight) providing judgments on arrival of new instances. A weighted majority voting method for weighting and combining classifier decisions is incorporated, after which a global prediction is derived. The weight of each classifier is adjusted depending on its accuracy in previous judgments.

3.2.3.3 Decay functions, Forgetting Factors and Half-Life

Sections 3.2.3.1 and 3.2.3.2 have demonstrated the shortcomings of time windows and ensemble classifiers. Another approach to learning evolving user interests is to simulate the natural forgetting mechanisms. Older information obtained through past search iterations is gradually *"forgotten"*. In a similar fashion, individual terms in the user profile vectors are weighted according to their age. Recent observations receive a higher weight than older data. Techniques to obtain such an effect have appeared in literature in various forms (Allan 1996, Koychev & Schwab 2000, Koychev 2001, Arampatzis & Van der Weide 2001). Most commonly, a linear *decay function* is employed to compute the term weights based on a forgetting factor. Similar to the time window size, a forgetting factor essentially defines the rate of learning. A high forgetting factor associates lower weights to older data and, thus, it exhibits much faster adaptation. On the contrary, slower forgetting implies a long-term information need and requires a lower forgetting factor.

Several functions have been proposed over the past few years to achieve such a time-dependent weighting effect. However, these techniques were developed in different contexts and applications, thus they lack a unified way to their influence to the term vector. For this purpose, a few researchers have introduced the notion of *half-life* of profile contents to heuristically describe the effect of the forgetting factor (Arampatzis & Van der Weide 2001). The half-life (HL) h defines the number of days after which the weight of a particular term t equals to half its original value. For instance, a function f with h = 20 reduces each term weight w_n in such a rate that after 20 days $w_n = \frac{w_n}{2}$.

Several implementations of gradual forgetting mechanisms have appeared in past research. Allan (Allan 1996) examined incremental relevance feedback in the context of adaptive retrieval and query expansion. A *slip factor* was employed to phase out old data and cope with *query drift*, where the focus of the user shifts over time. A set of experiments have been carried out to determine the most effective slip factor.

Koychev (Koychev & Schwab 2000) developed ELFI, a content-based web recommender system, that uses a liner forgetting function to cope with changing information needs. In ELFI, stored observations are weighted depending on their appearance over time.

Similarly, FilterIt (Arampatzis, Van der Weide, Koster & van Bommel 2000, Arampatzis & Van der

Weide 2001) constitutes an adaptive filtering system developed in the context of TREC-9 adaptive and filtering tasks. The authors employ a revised version of Rocchio's weighting method (Rocchio 1971) to incorporate a time-dependent weighting of terms. Also, a series of experiments have been performed with respect to the various HL and threshold settings. Although they discovered that the performance of the system peaks for a HL value of around 4 years, further analysis revealed that the structure of TREC topics greatly influences adaptation.

Fab (Balabanovic 1997) implements a recommendation service based on a combination of content-based and collaborative approaches. Profile terms are weighted based on the well-established TF-IDF formula, while a simple decay function is applied on a daily basis to reduce the weight of terms by 3%. Therefore, the rate of forgetting in their decay function equals to a HL value of approximately 23 days.

Observation of previous work in this area suggest that there is a great deviation in the half-life value in each forgetting mechanism implementation. The performance of the FilterIt system was assessed on the TREC adaptive track and is thus based on a lab-based environment. A long half-life setting of 4 years is far from optimal, since it suggests the presence of a relatively stable information need which may only exhibit a minor drift in the future. On the contrary, the profiling mechanism in Fab (Balabanovic 1997) appears to be more adaptive to changing concepts with a much higher learning rate.

In Colombus, I have opted for an online gradual forgetting mechanism to update the weight of profile terms as new examples arrive. Driven by the assumption that user search needs change rapidly, I used a half-life setting of 10 days, which was discovered empirically. The decay function implemented in Colombus is:

$$w'_t = w_t \cdot e^{-\frac{ln2}{HL} \cdot d}$$

where t is the current term, w'_t is the new term weight, w_t is the existing weight, HL is the half-life of the function and d is the number of days since the last appearance of the document containing term t. The application of this formula is independent of the term selection and weighting processes on each search episode and is directly applied to user profiles at the recommendation stage. An overview of the profiling and recommendation algorithms are provided later in this study.

3.3 Topic Overlapping

Research in the field of user profiling has mainly focused on defining models to detect user needs and adapt to concept drifts. Profiling algorithms are employed to create a snapshot of the user's search interests in the form of a profile. More sophisticated approaches represented the user's profile as a set of interests, while others based their research on a single-facet profile.

To our best knowledge, there is very little work done on hierarchical representations of user interests based on the vector-space model. I argue that users perceive their own profile as a set of possibly overlapping interests. For instance, an information retrieval researcher could be focussed not only on latest news in the field, but also for job openings in the area. There is an apparent relation between these interests. Such a connection between user interests is bound to influence the underlying representation of profiles. A hierarchy of user interests can be employed to represent relationships between user interests.

In Colombus, I have opted for a two-level hierarchy to represent user profiles more accurately. In effect, user interests can be part of the same category, or be in a world of their own. Overlapping interests have a parent-child relation where a larger scale topic, a *category*, contains several sub-topics. This hierarchical profile representation is explained in more detail along section 4.2.

Chapter 4

Profile Learning and Representation

4.1 Profile Learning

Personalization systems monitor user actions to create a content-based profile and keep track of their preferences. Profile learning is the process of long-term user modeling, as opposed to short-term session-oriented personalization via the creation of user profile. In this chapter, I demonstrate a high-level overview of the profile capturing process in Colombus and present alternative strategies used in previous literature.

Profile learning techniques consist of 3 main steps: (i) Relevance Feedback, (ii) Feature Selection and (iii) Profile Update. Relevance feedback gathering approaches have been extensively covered in section 6.4. The feature selection process refers to the extraction of representative terms from relevant documents. The idea behind feature selection is to assign an indicative weight to each term, representative of its worth in the term space. Subsequently, a cut-off function may be applied to obtain only a slice of the top-ranked terms and overcome the dimensionality problem (Baeza-Yates, Hurtado & Mendoza 2004). I propose a number of term selection techniques towards effective capturing of user profiles and present them in more detail along section 6.5.

During the update phase, the user profile is accumulated with information gathered during the previous

step. Due to the evolving nature of user needs and our need to adapt to changing concepts, the underlying learning scheme is of great importance. Many learning techniques were used as *base learners* in scientific literature. The main distinction that can be made in base learners is between *supervised* and *unsupervised* approaches. In the former, a set of examples is used to train the learning scheme. In fact, supervised techniques aim to deduce a classifier for incoming data, based on labelled training examples. These include explicit-based or domain-specific approaches, instance-based learners (e.g. K-nearest neighbour), support vector machines (SVM), decision trees and Naive Bayesian classifiers. On the contrary, unsupervised learning approach do not require any prior knowledge or labelled training examples for learning user profiles.

Information filtering applications based on explicit profile maintenance approaches were the first to emerge in past research. These systems required users to create their initial profile and explicitly amend it thereafter. Although the resulting profile are more accurate than those generated through implicit algorithms and techniques, they require a large amount of maintenance to keep them synchronized with current user's needs, which ultimately renders profiles out-of-date. In addition, explicit-based profile creation techniques have all the shortcomings of explicit relevance feedback, detailed in an earlier part of this research 2.1.1.

Decision trees algorithms have also been employed for the user profiling task (Payne, Edwards & Green 1997, Pazzani et al. 1996). A decision tree is a structure of nodes, each of them representing a particular decision or test that has to be carried out. Incoming data are classified by starting at the root node and working their way through the tree, until a leaf node is reached. In a similar fashion to decision trees, rule-based learning techniques use a set of rules to classify each example. Based on an induction policy, rules are extracted from training data and evaluated to classify incoming data. However, both decision trees and rule-based learning algorithms isolate object features, which leads to reduced precision compared to approaches that combine feature information (Godoy & Amandi 2005).

Naive Bayesian classifiers are among the most popular approaches to supervised learning of user profiles. This family of algorithms is based on Bayes theorem to apply a simple probabilistic classifier to a stream of incoming data. Given a complete profile, a naive Bayesian classifier can categorise incoming documents

to user interests (assuming that user profiles are multi-faceted). However, this technique can only be used for the classification task and is often combined with other approaches to implement the profile creation stage, as illustrated in (Carreira et al. 2004, Pazzani et al. 1996). In addition, Bayesian classifiers are required to be trained against a large set of data to increase their accuracy and achieve the best results.

In Colombus, I have opted for a K-nearest neighbour algorithm as the main component of the profiling scheme. This technique is a type of instance-based learning where computations are deferred until object classification. K-nearest neighbour algorithm has been extensively used in past studies due to its simplicity and the fact that it doesn't require any previous training. An object (e.g. web page) is classified and assigned to the class of its *closest* neighbour. Therefore, given a stream of incoming documents or terms, often obtained through relevance feedback, the closest neighbour N to D, the current object, is computed through a pre-defined distance function. At a later stage, object D is associated with neighbour N (often a term or document set). Several variations of this algorithm have employed a threshold T within the distance function. When the distance between the nearest neighbour and the incoming object D is less that the specified threshold, then a different path is triggered (e.g. D is discarded or created as a separate interest in the profile).

Although a great amount of research has been devoted to information filtering systems based on k-nearest neighbour to construct user profiles, the underlying distance function has always been the weakest link. In the context of information retrieval, the similarity between two objects cannot necessarily be described with simple mathematical formulas. Since functions such as cosine similarity or euclidean distance only take into account a current snapshot of profile data, incoming objects are sometimes miss-classified. For instance, the cosine similarity between term sets [car, purchase] and [buy, vehicle, drive, road] is 0, when the conceptual similarity is obvious. This effect has been observed during Colombus evaluation and is explained in more detail along section 10.5.

4.2 Profile Representation

This section describes the structure of user profiles and provides a thorough analysis of their components. It also explains the underlying reasons for implementing the current model and provides an overview of

other alternatives.

There are several ways to represent user profiles efficiently. Most recommender systems available have identified and satisfied the need to represent a profile as a set of multiple interests, independent from each other. In fact, these systems mostly differ in the way they represent each interest and, of course, the algorithms used to model and capture evolving user needs. Figure 4.1 illustrates an overview of the system's data model, showing all the profile elements and attributes.

More specifically, a profile consists of several interests each containing a representative name, such as "Computer Architecture", a vector of terms and a set of documents recommended by the system or explicitly bookmarked by the user. A profile also contains registration information consisting of a distinct string user id together with a password required to sign in the service, as well as a name and an e-mail address for each user. At first, a profile has no interests and eventually gets populated with user search requirements and preferences.

On the other hand, an interest must have a set of representative terms to describe the current search need, since, whether it was created implicitly or explicitly, it initially contains a non-empty vector of words. The document cache is a technique used to detect already seen information from new documents and is described in a subsequent section.

Similarly, document recommendations appear after some search iterations. Each recommendation is described by a document title, a search engine snippet, a summary generated by the relevance feedback algorithm, timestamps of when it was created, last access and modified and a Boolean variable indicating whether or not it constitutes a user bookmark. The timestamps variables are required by various system algorithms. The recommendation removal algorithm, discards all old, unvisited documents and thus depends upon the date a document was created and last accessed, while the last modification date of a document is expected during the creation of the home page.

The following parts of this section further investigate interest representation and possible alternatives considered during the design stage of this system. They focus on several ways to implement the term vector for each interest and illustrate the advantages and disadvantages of each approach.

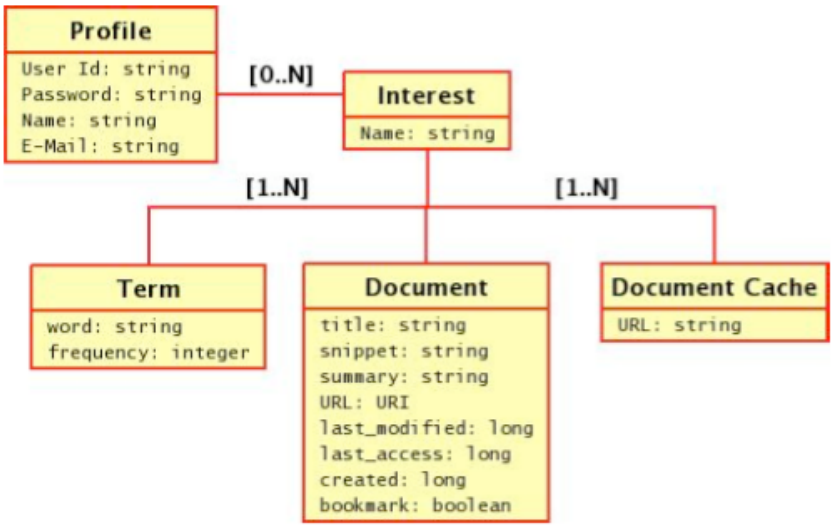

Figure 4.1: The data model

4.2.1 Weighted Semantic Networks

In past solutions, like ifWeb and SiteIf (Asnicar & Tasso 1997, Stefani & Strappavara 1998), user profiles were represented by weighted semantic networks. This approach represents terms and their context by linking nodes (words) with arcs which represent co-occurrences in some documents. However, this representation is rather unconventional and inefficient, since converting from semantic networks to a more suitable format requires some additional computations.

4.2.2 Binary Keyword Vector Approach

Another alternative is to use a vector containing a set of words to represent an interest. A word is relevant to an interest if it is contained in the data structure. Unfortunately, although this approach is easy to implement, it is mostly unusable due to the fact that algorithms must be able to distinguish between popular and rare words in an interest. Otherwise, document recommendations would depend on an increasingly growing set of words, thus rendering the system rather ineffective.

4.2.3 Weighted Keyword Vector Approach

Representing interests as a weighted keyword vector is arguably the most advantageous approach. Actually, Personalized Information Assistant is designed and implemented using this exact model. Interests are represented as a vector of weighted terms. Each dimension of the vector space represents a word and its weight. The weight of each term represents the number of times this word has been referenced. In fact, each referenced term's weight gets altered when the system appends a new set of terms to this profile. The system will increment all weights of terms appearing in the new set of words and the original interest. The profile generation algorithm is responsible for discovering the more appropriate interest and modifying it based on a given set of terms. Also, extracting popular words from an interest and performing interest comparisons, like a cosine similarity measure (Salton 1989) can be done very effectively and efficiently.

To optimize the profile representation scheme, a hash table was used to map each word to its weight. Each term constitutes a key to retrieve the related weight in the hash table. Due to the nature of this implementation, accessing terms and modifying entries can be done in constant time O(1), thus speeding up the profiling and document recommendation process.

Part III

Colombus Architecture

Chapter 5

User Interface

This chapter introduces the user interface design aspects and provides an insight of how the system would typically be used. It includes a detailed explanation of the most important design decisions and illustrates the use of the integrated symmetrization mechanism used to assist users in the complicated task of assessing document relevance.

Colombus is equipped with a search interface component, to allow users to discover information on the web. Also, the system features a personalized home page for each user, as well as a profile editor and a bookmark manager, available to incorporate explicit feedback. The home page facilitates access to recommendations by displaying documents discovered by the system that might be of interest for the user. Additionally, the profile manager allows users to modify their profile by adding, amending or removing interests. However, explicit interaction with this profile editor feature is not mandatory. Finally, the bookmark manager works in a similar way to the home page, but only displays those documents that have been explicitly bookmarked by the user for convenience. A more detailed analysis of these components is presented in the following sections.

5.1 Interaction with the User Interface

Colombus user interface comprises of several components to pro-actively capture user profiles and recommend document relevant to their drifting needs. In this section, I document the ways searchers can interact with the user interface, by summarizing and presenting them in table 5.1. The *"System Component"* column indicates the user interface part that this action can take place, while column *"Component Affected"* shows what are the implications of each action in the underlying algorithms.

Interaction	**System Component**	**Component Affected**
Login	Navigation	*None*
Logout	Navigation	*None*
Visit profile manager	Navigation	*None*
Visit bookmark manager	Navigation	*None*
Visit personalized home page	Navigation	*None*
Issue search query	Search interface	*None*
Follow search result	Search interface	*Profile learning: Shows user's implicit interest*
View search result summary	Search interface	*Profile learning: Shows user's implicit interest in voting schemes*
Bookmark search result	Search interface	*Profile learning: Shows user's implicit interest in voting schemes*
Follow recommendation	Personalized home page	*None*
Remove recommendation	Personalized home page	*None*
View recommendation summary	Personalized home page	*None*
Follow bookmark	Bookmark manager	*None*
Remove bookmark	Bookmark manager	*None*
View bookmark summary	Bookmark manager	*None*
Add interest	Profile manager	*Profile learning: A new interest is added, so new information can flow and ammend the underlying temrm vector*
Remove interest	Profile manager	*Profile learning: An existing interest is removed, so it directly affects flow of information*
Amend interest	Profile manager	*Profile learning: An existing interest is changed (terms added/removed), so it directly affects flow of information*

Table 5.1: Interaction with the user interface

5.2 Registration / Login Interface

Any system aiming containing personalization and recommendation features includes login functionality to allow users to view and maintain their own recommendations. Colombus follows the same approach and integrates a centralized login mechanism presented to each user that visits the system. In fact, no action is possible if the user decides not to login into the system. A new user can register via the similar interface presented in the screenshot below.

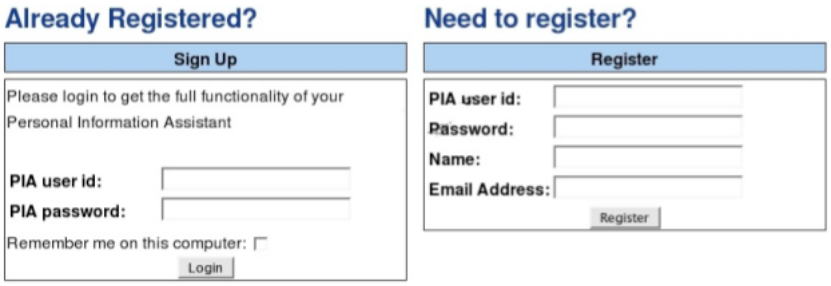

Figure 5.1: Registration / Login interface

5.3 Search Interface

With searching central to every information seeking environment, it was important to design the search engine interface with ease of use and simplicity in mind. The internal search engine empowers users with the ability to discover information on the web. It is based on a search component that communicates with Google to retrieve online documents. A more detailed analysis of the architecture and implementation of the search module is presented in section 6.3.

Furthermore, the search engine result layout constituted a primary concern when designing the various user interface components. In order to maintain external consistency, it was decided to present the search results by mimicking Google layout, since it constitutes the most successful commercial search engine, so users of Colombus will feel familiar with the interface. Also, since the system communicates with Google, the retrieved set of results will be the same for Colombus and Google. Therefore, it makes sense

to provide a similar interface to give users the impression that they are searching through the popular information engine. As illustrated in figure 5.2, documents are described as in Google with the addition of the summary presentation provided when the mouse hovers over each link.

Another feature provided in Colombus' search interface is the ability to bookmark a document potentially for future exploration or simply for convenience. After the completion of a search iteration, the system will analyze the implicit data gathered through the user's interaction with the system and will modify his profile accordingly. Bookmarked documents are added as recommendations in the interest associated to the current search iteration, providing users with the opportunity to read them at some point in the future.

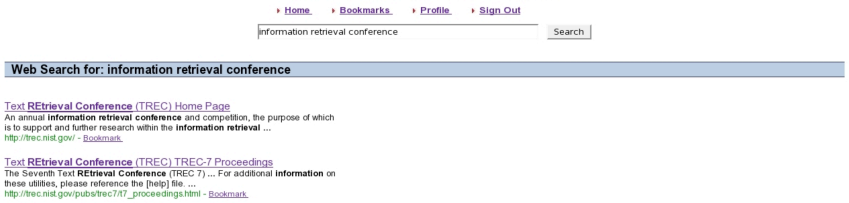

Figure 5.2: The search engine interface, illustrating a sample results page

5.3.1 Document Summarization

The presentation of search engine results in the web have illustrated a constant non-evolving trend for many years (Joho & Jose 2008). Basic textual representations of results have been restricted to various forms of hyperlink and title lists, when the benefit from augmented document surrogates has been illustrated in several studies in the past (Hearst & Pedersen 1996, Chen & Dumais 2000, Joho & Jose 2008). Such incomplete presentation of search results can create confusion with users during the crucial document judgment stage.

In addition, short queries in web search environments and the uncertainty of users with regards to their actual search needs further amplifies this problem. Users are expected to make informative decisions based on a list of vague search results, while starting with very little information about the topic. In effect, incomplete document surrogates make things worse since the lack of proper document descriptors

forces users to prolong their search episodes by visiting a large amount of documents in order to find the information they need.

We have employed a document summarization component, designed and implemented in (White, Ruthven & Jose 2002*a*) to improve the search experience through Colombus. This component creates query-dependent summaries by parsing each document in the search results and top-ranking sentences based on their similarity to the user's query. Summaries in Colombus are presented in the form of a popup box activated on mouse-over or through the summary link. An example of the summarization component in action is presented in figure 5.3.

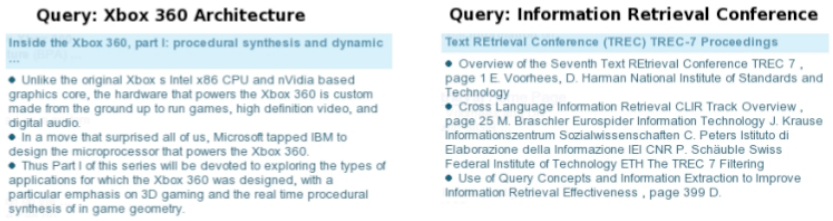

Figure 5.3: Two examples of query-biased summaries of search results

5.4 Recommendation Portal

The personalized home page of each user constitutes a central hub in Colombus' user interface. In some sense, it has similar functionality to a portal, where people can view documents recommended by the system with respect to their interests. As shown in figure 5.4, the page provides facilities to delete or bookmark documents, as well as indirect manipulation of user facets and interests. Trying to change an existing interest through this page redirects the user to the profile manager. Features and functionality of this component are discussed in further sections. Containing all these navigational and operational features and functions, the home page works as a co-ordinating hub for other activities.

Recommended documents are described by their title, search engine snippet, summary on mouse-over events and the number of days since they were modified. This last piece of information is not always

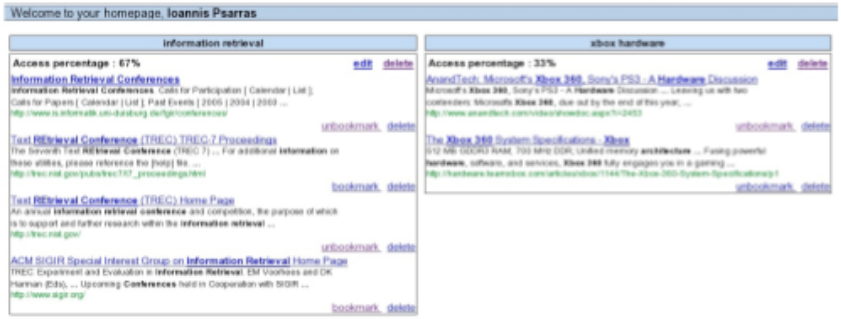

Figure 5.4: The personalized home page, displaying additional documents discovered by the system

available since it is discovered in HTML META data field, which is not always updated by designers after each page modification.

As discussed previously, Colombus contains all the required features to allow users to locate previously seen information. The implemented algorithms comprising the system core purge unvisited recommendations after some days, in case the user shows no implicit interest. Implicit interest in a document can be shown through the bookmark facility. More details about how these algorithms work are provided in sections 6.2, 6.3 and 6.4.

In addition to the search results page, bookmark facilities are also ordered in this case, thus allowing users to explicitly inform the system that they are interested in a document, without actually having to view it. These documents can be reviewed and modified in the bookmark manager.

5.5 Document Bookmark

Users can bookmark interesting information either during their search iterations, or when browsing the recommended document in their home page. The system's interface features a bookmark manager component to allow users to re-locate and view their bookmarked documents and to facilitate information access. The look and feel of this component, as figure 5.5 demonstrates, is very similar to the home page, in order to enhance internal consistency and navigability of the web site.

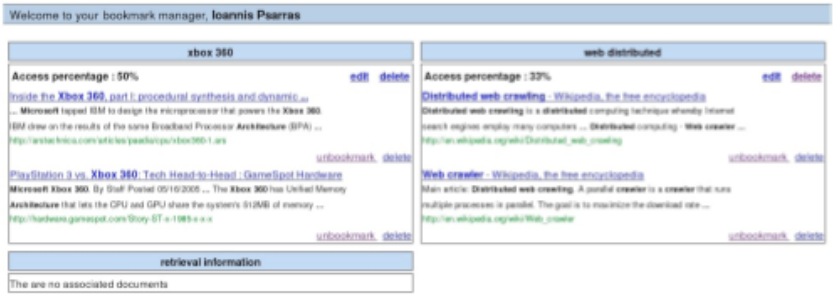

Figure 5.5: Document bookmark in Colombus

5.6 User Profile Management

A major component of the system is the profile capturing scheme. Interests can be entered explicitly, but it is also possible to capture user interests through their interaction with the system (Kelly & Belkin 2001, Kelly, Diaz, Belkin & Allan 2004, Morita & Shinoda 1994, White et al. 2002*a*). The profile manager formulates the interface for explicit interest addition, modification and removal. Each profile constitutes of multiple interests. An interest is a set of terms, essentially words describing the interest, distinguished by a representative title. A more thorough analysis of profile representation is provided in section 6.1. People can add their own interests explicitly by completing the name and terms fields. Similarly, a user can modify an existing interest by altering its terms and submitting the changes for review. Last, but not least, the profile manager interface is also accessible through the home page, by choosing to edit or delete an interest, to enhance the navigational features.

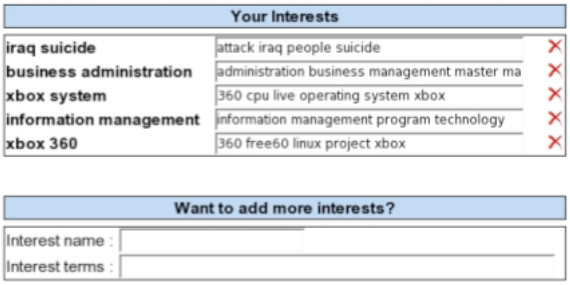

Figure 5.6: The profile manager

Chapter 6

System Design & Implementation

6.1 Introduction

The Colombus system is situated on a server and interacts with users through Java's servlet technology (J2EE). User queries and other interaction data is captured and processed at the server. The document indexing and querying have been realized by employing the Google API framework. After each search iteration, the user's profile is updated to include the latest information. As detailed in 6.4, relevance judgments are provided in the implicit form of click-through data and bookmark actions.

The remainder of this chapter is divided as follows: Section 6.2 presents a high-level overview of the user profiling and recommendation process, while an example scenario is provided to demonstrate this process more effectively. Sections 6.3 illustrates the search module and term selection algorithms respectively, while section 6.4 covers the relevance feedback policy in terms of the sources of information and the procedure to track searcher actions with the user interface. The term weighting approaches are presented in greater detail along section 6.5. Then, an in-depth analysis of the recommendation procedure is presented in section 6.7, while last but not least section 6.8 outlines several design and implementation decisions made during the development stage of Colombus.

6.2 Process Overview

As detailed in section 3.3, a hierarchy of topics represents the closest snapshot to the way user perceive their search needs. I have opted for a two-level hierarchy for the user profile. In essence, topics in the same, broader, area are clustered together based on a two-pass clustering scheme.

After a user A issues a query Q, Colombus fetches the results from Google and presents them to the screen for further examination. This part is reflected on the *search module* which is responsible for parsing user queries and retrieving results from the web. Then, user A will normally interact with the system by deciding to read some documents, view summaries of others and maybe follow up or bookmark them in the manner shown along chapter 5. Colombus will collect implicit relevance feedback by analyzing the user interaction with the system and a relevant document set will be derived. This part of the process is driven by the *relevance feedback policy* as well as the underlying *term selection algorithm*. The top-ranked terms are used to formulate a vector V of term-weight associations. At this point, the clustering algorithm is triggered to update the user's profile, based on a pair of thresholds, GT (i.e. group threshold) and DT (document threshold).

1. User issues query Q
2. User interacts with the result set
3. System collects relevance judgments in an implicit manner
4. System weights terms based on weighting scheme and generates a vector V with top 7 terms
5. System updates user profile
   (a) Compare V against the set of profile facets (i.e. interests)
   (b) Find the closest interest I
      i. If the similarity between vector V and interest I is above threshold GT and below DT
         A. Create new interest from V and add it in the same cluster as I
      ii. If the similarity between vector V and interest I is above threshold DT

A. Update interest I by incrementing its contents with terms from V

iii. If the similarity between vector V and interest I is below threshold GT

A. Create new interest from V

Figure 6.1 shows a general overview of the approach used.

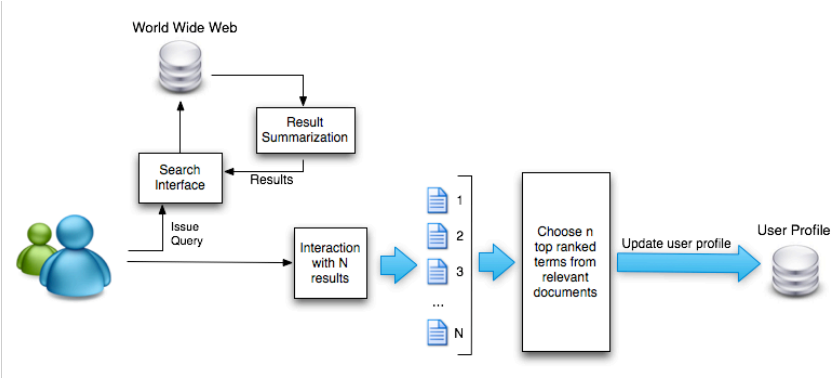

Figure 6.1: Colombus profiling process overview

6.3 Search Module

Since searching is central to any information seeking environment, it was important to design the architecture of the search module with both extensibility and flexibility in mind. The search module interfaces with Google to retrieve a result set. This is done via methods provided by Google API, allowing developers to integrate the popular search engine in their applications. After each user search, the module retrieves a result object and applies a port-processing algorithm to convert it to the required structure.

The conversion of HTML to plain text, the summarization of search results, along with the creation of objects readable by Colombus, are among the actions accommodated in the result post-processing algorithm. Although search results are retrieved rapidly from Google, the summarization process causes a significant amount of delay during the post-processing stage resulting to an approximate *"search to display"* time of 3-5 seconds.

6.4 Relevance Feedback

When the search process has completed, the N most relevant results are displayed. N describes the amount of documents included in each search result page and has been set to 10. At this point, we start tracking user actions with the search results. Although the parent actions are mainly click-through and mouse movements, several sub-interactions have been pre-defined, such as *hover-over-summary*, *visit-through-title* and *bookmark*. Section 5.1 presents the pre-defined types of interaction in greater detail.

When the searcher interacts with a search result, the system translates his action in a stream of weighted words extracted from the document surrogate. The term weighting process is detailed in section 6.5. Depending on the underlying term weighting scheme, each user interaction with a search result is translated to a distinct set of words. For instance, a *hover-over-summary* action will trigger the weighting of words in the result summary, when a voting-based scheme is employed. A voting term weighting scheme employs an interaction-sensitive approach to term weighting, where user actions are assigned a confidence factor. Each of these word sets represents a piece of information towards the identification of the user's search interest. In effect, Colombus performance, from adaptive profiling to document recommendations, is analogous to the quality of the information extracted at this stage. Therefore, we can capture and analyze the performance of relevance feedback and term weighting by observing the quality of user profiles. There has been a great focus on this aspect during the simulation study (Chapter 7). At the end of this stage, word sets are merged, by accumulating the term weights, to generate a master set of N word-score pairs.

6.5 Term Selection

The term selection process entails the selection of words that describe the user's information need in the best way possible. Term selection constitutes the core component of query expansion algorithms aiming to extract those informative terms from relevance feedback judgments during the past search iteration. Also, several applications that re-rank results based on user past search iterations employ term selection to build their profiles, in the same way that personalized recommendation services discover relevant

documents to reduce the cognitive load from users and eventually facilitate their search experience. In effect, a term selection algorithm is a vital cog in any user profiling system.

I have developed a number of term selection schemes to support the profiling process. In this section, I introduce the notion of *voting* term selection schemes and differentiate them from classic approaches. I will also present the details of 4 different strategies to term selection supported by implicit relevance feedback collection in the form of click-through data.

6.5.1 Voting vs. Non-voting schemes

The usual course of action in term selection algorithms is as follows:

1. User issues query
2. System collects relevance feedback in an implicit or explicit manner
3. System weights terms based on some formula for those objects judged as relevant
4. System selects the n top-ranked terms

The above methodology has worked very well over the past years, but excludes user interface actions apart from simple relevance feedback collection mechanisms. Relevance feedback collection serves only to distinguish relevant objects in the information space, irrespective of whether object ratings have been collected from user interaction with the UI. I argue that a heuristic-based term selection scheme where different user actions have different implications can result in an optimized term vector.

In a *voting* scheme, each user interface action is associated a different weight, while a different set of terms is processed for each UI interaction. These actions accommodated in the system as part of our voting schemes are:

1. Follows search result
2. Views summary

3. Bookmarks result

I argue that each of the above actions must be associated with a heuristic weight $a_n$as an indicative worth, to compensate for the fact that each interaction with the user interface has a different meaning. A view of the summary is not as strong as the bookmark of a search result. Furthermore, a set of term vectors is linked to each of the above actions, allowing parts of the document surrogate to *vote* for their terms. In our case, a click on the summary allows voting for terms contained in the summary, bookmarking a search result triggers votes for an information object. In a similar fashion. searchers follow up a link to a search result influenced by the document's title and snippet. Hence, these parts of the surrogate will vote for their terms. In the rest of this section, I present an example to further illustrate the effect of a voting scheme against that of classic term selection.

Let a user A issues a query about some topic. Let 10 search results, named r0 to r9 with each result object consisting of a title (with term vector t_n), a snippet (with term vector sn_n), a summary (with term vector sum_n), a content (with term vector c_n) and a bookmark feature. Finally, assume the presence of an implicit feedback collection policy based on click-through data from the current search iteration and user A proceeds in the following actions:

- Views the summary of r_0
- Follows the title link to r_1
- Follows the title link to r_2
- Bookmarks r_5

Then, in a voting scheme, the vector V of terms from the past search iteration is:

$$\mathrm{V} = a_2 \cdot sum_0 + a_1 \cdot (sn_1 + sn_2 + sum_1 + sum_2) + a_0 \cdot (c_5 + sum_5 + sn_2 + t_5)$$

The weights a_n are set empirically and are fixed to 0.8 for summary, 1.0 for title and 1.2 for the bookmark feature. Term weighting in each term vector is accommodated based on the underlying weighting schemes.

The terms with the highest overall weight are considered to best describe the information viewed by the searcher and are therefore further processed during the profiling process. The rest of this chapter presents the weighting strategies implemented in Colombus.

6.5.2 Binary term weighting

In binary weighting scheme, the term is associated with a weight of zero or one, depending on whether it is present in the document. The weights assigned to individual terms in each search result are accumulated together to produce a set of word-score pairs. In effect, the weight of each term t_i is equal to the number of occurrences in the relevant document collection (i.e. the documents classified as interesting after the relevance feedback process). The terms with the highest overall vote are considered to describe better the information viewed by the user (i.e. those terms that are present most often across all representations) and can be used to approximate searcher interests.

For example, consider documents D1 and D2 , with D1 containing terms T1 appearing once and T3 appearing 5 times, while D2 contains terms T1 and appearing once, T2 appearing 6 times and T3 occurring 2 times. The term extraction table would be:

	D1	**D2**	**Total Weight**
T1	1	1	2
T2	0	6	1
T3	5	2	2

Table 6.1: Example of term extraction table for the binary term weighting scheme

6.5.3 Term frequency (TF)

This model employs the widely used term frequency (TF) measure in the term weighting process. TF is a statistical measure illustrating the importance of a term in the document collection. The importance increases proportionally to the number of times a word appears in the document.

The term frequency in the given document equals to the number of times a given term appears in that document. This count is normalized to prevent a bias towards longer documents (which may have a higher

term frequency regardless of the actual importance of that term in the document) to give a measure of the importance of the term t_i within the particular document.

Let's again assume the existence of two documents as described in the earlier example. Also, the total number of terms for each document is shown in angular brackets. The term extraction table would be:

	D1 ***[100]***	**D2** ***[80]***	**Total Weight**
T1	1	1	0.0225
T2	0	6	0.0750
T3	5	2	0.0750

Table 6.2: Example of term extraction table for TF weighting scheme

6.5.4 Term Frequency - Inverted Document Frequency (TF-IDF)

The TF-IDF scheme is a vector-space model, taking into account the term frequency TF in a document, as well as the term scarcity IDF in the whole collection. TF-IDF is still used with great success in search engine and information filtering applications.

The inverse document frequency (IDF) is a calculation designed to make rare words more important than common words. Traditionally, each component weight is assigned using the formula:

$$tf - idf_t = tf_t \cdot log\frac{D}{D_t}$$

where tf_t = term frequency of term t, D = number of *relevant* documents in the collection and D_t= occurrences of term t in *relevant* document collection.

Taking the same example as above, the total number of terms in each document is shown in angular brackets. Then, the term extraction table would be:

6.5.5 WPQ

I have also employed the popular WPQ (Robertson 1990) method to assign term weights. The equation for WPQ is shown below:

	D1 *[100]*	D2 *[80]*	Total Weight
T1	1	1	0
T2	0	6	0.0583
T3	5	2	0.0408

Table 6.3: Example of term extraction table for TF-IDF weighting scheme

$$wpq_t = log\frac{(r+0.5)*(N-n-R+r+0.5)}{(n-r+0.5)*(R-r+0.5)} \cdot (\frac{r}{R} - \frac{n-r}{N-R})$$

where N = number of document in collection, n = number of documents in collection containing term t, R = number of relevant documents and r = number of relevant documents containing term t.

WPQ is based on probabilistic distribution of terms in relevant and non-relevant documents. As relevant documents and terms change between search iterations, their underlying weights are likely to change too. The example presented previously is not really applicable to WPQ term weighting since the small collection size causes a division by zero arithmetic operation and the relevant table has, therefore, been omitted.

6.5.6 Jeffrey's Conditioning

Jeffrey's conditioning attempts to weight terms by taking into account the user's interaction with the search results. In essence, terms are weighted according to the underlying type of implicit evidence (see section 5.1). The approach I used in this revision is based on the one proposed by White (White et al. 2005) but has been altered so it is applicable in the context of document recommendations.

The idea behind Jeffrey's conditioning is that each interaction with the user interface indicates the user's confidence about the current document's relevance. For instance, I can assume that bookmarking a document denotes much stronger evidence than simply hovering over its title. The confidence for each particular user action I is denoted as c_i. In Colombus, there are three possible user actions on document surrogates (presented in decreasing order of user confidence): bookmark, click-on-title, view summary. The confidence of each user action is empirically assigned a pre-defined weight. Document bookmark boosts the underlying term weight with a confidence of 1.2, click-on-title leaves term weights unchanged (confidence of 1), while viewing the document's summary slightly decreases weights (confidence of 0.8).

Therefore, acting on a document representation pi creates new evidence for the underlying terms. Jeffrey's rule of conditioning is then employed to update the probability of term t using the following formula:

$$\text{P'(t)}=\sum c_i.\left[P_i(t=1|p_i)\frac{P'_{i+1}(t=1)}{P_i(t=1)}+P(t=0|p_i)\frac{P'_{i+1}(t=0)}{P_i(t=0)}\right].P(\text{t})$$

where P(t=1) is the probability of observing t, P(t=0) the probability of not observing t *and* c_i is the confidence of action *i*. *Similarly,* $P(t)$ is the probability of term t being relevant based on the probability distribution P over term space T and is defined as $\text{P(t)} = \frac{ntf(t)}{\sum ntf(t)}$ where ntf(t) is the normalised term frequency of term t in the term space T. The effect of this variation of Jeffrey's conditioning is similar to the result of our voting schemes.

6.6 Profile Learning

The profile learning algorithm is responsible for accommodating the information extracted during relevance feedback into the user's profile. Users often have search needs very close in terms of subject area. Often, there are several aspects regarding the same topic. The underlying user interest for a query *"car mechanics*" is, to a certain degree, different than one about *"car rentals".* The profiling algorithm should be able to distinguish between different facets on the same topic. In this case, information about car rentals and car mechanics have to be accommodated as two distinct interests in the user's profile, such that new document recommendations are catered separately. As detailed in section 4.2, I have opted for a two-level interest hierarchy to represent *groups of interests.*

Users often have search needs very close in terms of subject area. Often, there are several aspects regarding the same topic, such as information retrieval masters and information retrieval techniques. The algorithm should have an adequate degree of sophistication to distinguish different facets on the same topic. In Colombus, the profile learning algorithm creates groups of interests. Depending on their degree of similarity, the algorithm recognizes whether an existing interest has to be augmented, or a new one has to be created. However, in the latter case we also track whether an existing *topic* can be employed to store new information as a child interest.

Profile learning is realized through a two stage k-nearest neighbour approach. As it represent potentially unseen information, the term set S calculated at the end of the term extraction and relevance feedback process is directly related to this algorithm. At the first stage, all interests in the profile are clustered to create a two level tree hierarchy. At the top level we have *topics*, while the *interests* themselves lie in the lower level. An overview of the Colombus profile representation approach is presented in figure 6.2.

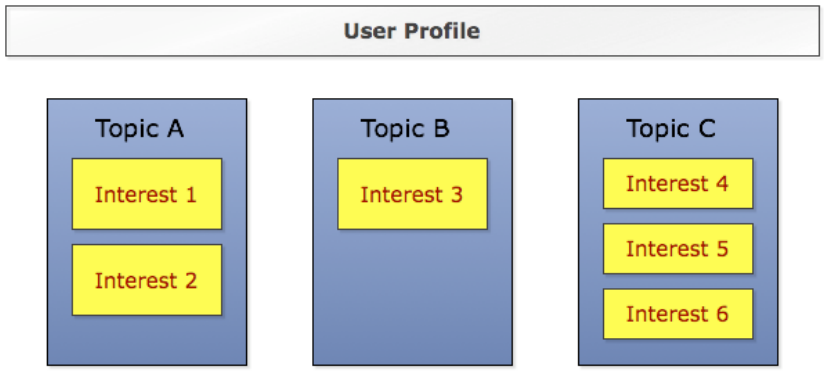

Figure 6.2: Profile hierarchy in Colombus

As detailed in section 4.2, this is the standard representation of user profiles within Colombus. Then, each topic T is compared to the term set S and its similarity value is calculated. Finally, as long as the set S is *adequately similar* to its most similar topic T, then T is updated to contain S. A topic is *adequately similar* to term set S when their similarity value is above a pre-defined threshold C. In that case, the nearest neighbour algorithm is re-run on term set S and the children of T, i.e. the interests in T. The similarity matching measure used throughout this process is the robust cosine coefficient (Salton 1989).

Therefore, a pair of threshold values define the path new information take before being accommodated in the user's profile. The threshold employed at the first stage of the algorithm will be refereed to as *cluster threshold (CT)*, while its counter-party in the second stage is called *document threshold (DT)*. An overview of the profile learning algorithm is presented below, where the similarity between two objects (i.e. topics or interests) is denoted as Similarity[A, B]:

- Cluster the interests in the user's profile to form topics (i.e. groups of interests)
- Find the closest topic T to the given set of terms S
- If Similarity[T, S] $<$ CT, then create a new topic Q and accommodate S as a new interest in Q.
- Else, find the closest interest I in T to the given set of terms S
- If Similarity[I, S] $<$ DT, then create a new interest Q as a child of existing topic T.
- Else merge S in interest I.

6.6.1 Clustering Algorithm

The clustering algorithm used is an implementation of single pass clustering using cosine coefficient as the distance function. This method creates a partitioned dataset as follows:

- Make the first object the centroid for the first cluster.
- For the next object, calculate the similarity, S, with each existing cluster centroid, using the cosine coefficient.
- If the highest calculated S is greater than some specified threshold value, add the object to the corresponding cluster and re determine the centroid; otherwise, use the object to initiate a new cluster.
- If any objects remain to be clustered, return to step 2.

Among the advantages of single pass clustering is that it requires only one pass through the dataset, so the time requirements are typically of order O(NlogN) for order O(logN) clusters.

6.7 Recommendation Process

The relevance feedback and user profiling process is executed independently from document recommendations. At this stage, the hierarchical structure of user profiles has no effect to the quality or quantity

of document recommendations. The algorithm examines user profiles and issues a web search for each of the existing interests. Then, the top N ranked items for each web search are added to the underlying interests and presented as new document recommendations in the recommendation portal.

In essence, the recommendation component is an add-on to the profile learning algorithm. As no document recommendations can be effectively realized without a long-term user profiling component, our simulation and user experiments (chapters 7 and 9 respectively) focus on the learning process in terms of the quality of profiles and adaptivity capabilities for drifting user needs.

6.8 Implementation

6.8.1 High-Level Architecture

This section provides a detailed analysis of the system's architecture and examines the vari- ous technologies and programming languages used to develop and deploy Personal Informa- tion Assistant. Also, it focuses on the benefits of this architecture, as well as, the reasons for choosing the programming language environment. Although this section is not related with the area of information retrieval, it allows us to examine the impact of the current architecture to the system as a whole. The system is situated on a server and interacts with users through Java's servlet technol- ogy. It is based on a three-tier architecture to exploit the scalability and flexibility benefits of this model. Section 6.8.2 below provides a more detailed description of three-tier architectures. The main servlet engine together with the system's core are driven by an Apache Tomcat server. In addition, a remote data repository is used to implement part of the internal persistence manager, responsible to load and save user profiles. The server is also in control of the system's core that integrates and drives all the algorithms required to service users.

6.8.2 A Three-Tier Model

The three-tier model is a client-server architecture in which the user interface, process logic, data storage and data access are developed and maintained as independent modules, most often on separate platforms.

The three-tier model is a well-established software architecture and design pattern. The process logic may consists of more than one modules running on a centralized server. The user interface runs on a desktop PC or workstation and uses a web browser, in our case, to access the provided services. Finally, the data storage is usually realized by a database application. In Colombus, the internal persistence manager uses a database to store and access user data. The three-tier architecture is intended to allow any of the three components to be upgraded or replaced independently as requirements or technology change. The main benefit of this architecture is the separation of the process logic from presentation logic. The process logic engineer need not worry about formatting the output; the programmer developing the Web page need only be concerned with the output data that will be passed the page. In addition, this separation makes it possible to change the presentation layer, the user interface, without modifying the code that implements the process logic, thus increasing its reliability and robustness. Apart from the usual advantages of modular software with well-defined interfaces, a three-tier architecture improves radically the portability of a system. For example, Colombus can be accessed with various web browsers irrespective of their features or the operating system they run.

6.8.3 JAVA Servlet Technology

A servlet is a JAVA application that runs in a Web server or application server and provides server-side processing such as accessing a database and e-commerce transactions. Widely used for Web processing, servlets are designed to handle HTTP requests and are the standard Java replacement for a variety of other methods, including CGI scripts, Active Server Pages (ASP) and proprietary C/C++ plug-ins for specific Web servers (ISAPI, NSAPI). Having access to the full range of the powerful JAVA API constitutes the most important advan- tage of servlet driven applications. Therefore, servlets support several capabilities that are difficult or impossible to accomplish with regular scripting language. Servlets also introduce the notion of state from request to request, simplifying techniques like session tracking and caching of previous computations.One of the main sources of vulnerabilities in traditional scripting language, like CGI, is the fact that the programs are often executed by general- purpose operating system shells. This approach introduces security concerns for applications developed in these languages. On the

contrary, servlets suffer from neither of these problems. Even if a servlet executes a system call to invoke a program on the local operating system, it does not use a shell to do so. And, of course, array bounds checking and other memory protection features are a central part of the Java programming language.

6.8.4 Profile Persistence

Information persistence is a major requirement for our system. Profile lifetime extends beyond a single use of the system and recommendations always have to be available for the user. All these issues were dealt with by introducing profile persistence to extend information availability beyond the lifetime of JAVA Virtual Machine. Profiles are maintained in the disk as XML files taking advantage of this technology's flexibility and well-defined structure. An example XML schema for a user profile is provided in appendix B.1.

Since profile retrieval and saving are done in real-time, operation speed was a major concern when designing the persistence module. I have employed Java Architecture for XML Binding (JAXB) API to implement profile input/output operations. JAXB parsers are based on an object mapping between XML entities and JAVA classes.

A major issue during the design of the persistence manager was the real-time update of profile data for the current user. Disk operations are very time-consuming, so saving the state of a profile must be done as infrequently as possible. Taking this into consideration, an attempt to optimize the system was made by storing profile data in memory and updating the XML structure on user sign out. In a similar fashion, user profile information is loaded only when he signs in. Therefore, read/write operations have been minimized to 2 operations per session, parsing and updating the user's profile. The remainder of this section presents some statistics on storage and time requirements of the XML-based representation scheme.

The statistics presented in this section are calculated by averaging the results of the simulation and user studies. According to table 6.4, the size of user profiles is closely related to the number of interests, averaging 24.66 kilobytes, which is considered to be within the desired boundaries. In addition, decent

profile loading times were achieved, averaging a total of 157 ms to retrieve user information from the disk. Saving time was found to be almost negligible.

	Interest Size	Recommendation Size	Profile Size (in kb)	Profile Loading Time (in ms)	Profile Saving Time (in ms)
Minimum	1	5	2.4	157	5
Maximum	17	63	53	162	99
Average (Std. Deviation)	7.33 (4.39)	30.39 (17.55)	24.66 (15.19)	161.29 (1.84)	44.77 (47.7)

Table 6.4: Statistics on storage and time requirements of the XML-based representation scheme

Part IV

System Evaluation

Due to the increasing needs from context-based information retrieval tools, the sophistication of user profiling algorithms has increased by several orders of magnitude. Although, the parameters supporting these techniques are often numerous, most simulation experiments skip the tuning stage (Arampatzis, Beney, Koster & Van der Weide 2000). To our best knowledge, there has been no large-scale study to evaluate the effect of varying influencing parameters on a profile-aware retrieval tool.

Furthermore, there archapters no established approaches for evaluating implicit feedback based search tools, such as adaptive recommendation assistants. Simulation-based evaluation methodologies exclude the users from the evaluation stage, but results can be collected in large quantities (White et al. 2005). On the other hand, user-centered experiments have been criticised for the difficulty involved in setting up a proper evaluation study (Mostafa et al. 2003), while their short duration makes them inappropriate for benchmarking systems that deal with long-term information needs. I propose a hybrid evaluation framework based on a series of simulation runs and user-centered experiments.

The following chapters are organized in the following structure: Chapter 7 describes the methodology of the simulation experiment, covering the design of the searcher stereotypes, as well as a series of scenarios employed to benchmark several aspects of our system. Chapter 8 summarises the results obtained during the lab-based simulations and optimizes the algorithmic configuration based on observed trends. On the other hand, chapter 9 presents our proposed task-based, user-centered experimental strategy, while in chapter 10 I analyze the outcomes of this study.

Chapter 7

Simulation

7.1 Introduction

The simulation experiment is designed around the concept of searcher simulations (White et al. 2005) where the search behavior of typical user stereotypes is simulated to obtain a more realistic set of results. A set of user stereotypes with pre-defined search behaviours constitute our "*experimental subjects*" throughout this simulation study. The aim is to obtain an indication of the system's performance by simulating typical search interactions. A series of parameters are varied between experimental runs to gather a large set of results under different circumstances. The remainder of this chapter presents the evaluation methodology in terms of experimental scenarios (section 7.7), simulation components (section 7.4) and statistics(section 7.9), as well as the design and implementation of searcher stereotypes (section 7.3).

7.2 System, Collection and Topics

Terrier information retrieval platform (Ounis, Amati, Plachouras, He, Macdonald & Johnson 2005) has been emoloyed to index the TREC AP collection. The Terrier platform was configured to eliminate stop

words and apply stemming during the indexing stage. Also, BM25 (Ounis et al. 2005) was employed as the underlying retrieval model, among those implemented in the Terrier platform.

The TREC AP collection is derived from proprietary AP news data and comprises 242,918 documents/stories with an average length of 286.36 words per document. Also, we worked with TREC topics 251-300 and took each query from the short title field. For each query, the terrier system retrieves 100 documents on which the evaluation procedure is applied. The size of the retrieved set of results was intentionally decided to be rather large to avoid queries with no relevant documents in a smaller result set.

In spite of the TREC effort to clean up, the TREC AP collection still contains a small amount of noisy data, like fragmented text and missing sentences. Also, the relevance assessments on the TREC AP collection were done by professional journalists, which may produce biased evaluation results assuming a more experienced set of searchers.

7.3 Designing Searcher Simulations

This experiment is largely based on the concept of searcher simulations. The idea of such techniques is to simulate the interaction of searchers with the experimental system to obtain more realistic results. Search behavior is closely related to human dynamics and often psychology. Although these research areas are not directly related to our study some basic concepts to be included in our experiments. In order to accurately simulate search behavior, we must first define the factors that affect it. *User patience*, *experience* and *previous knowledge* have been identified as influencing factors of search behavior and combined to build user models for our experiments. The following sections cover these search behavior characteristics in detail and present implementation decisions in the context of search simulations for evaluation of information filtering systems.

7.3.1 User Patience

The *patience* of the user represents how deep a user usually goes down the search results looking for relevant documents. Some searchers are more content to go deep into the search results, while others will look at the top 3 or 5 documents only. This patience measure is directly related to the position of search results: the deeper we dive into search results the less the user's patience gets.

To our best knowledge, integrating a patience measure in a search simulation algorithm has never been examined before. Since the patience p_i of a user illustrates his eagerness to dive deeper in the search results, it can be defined as the number of documents he is willing to examine before he stops the current search iteration. For example, a searcher with $p_i = 4$ will visit 4 documents in each search iteration.

7.3.2 User Experience

Another characteristic that influences search behavior is *user experience*. During a search iteration, people observe a large number of documents, both relevant and not (or less) relevant. Arguably, more experienced searchers can make more accurate relevance judgments based on their previous experience.

The experience of a user can be presented by noise in the search results. A highly noisy environment represents the confusion of amateur computer users. I simulate the effect of a noisy environment by introducing a number of irrelevant documents in the relevance assessments. In effect, the user's experience, e_i = percentage of irrelevant documents to be injected.

7.3.3 Previous Knowledge

Similar to the search experience characteristic, a user's knowledge on a particular area affects their overall search strategy. A searcher with greater knowledge on a topic will look for documents more focused to this particular area. For example, a computing science professor is unlikely to look for documents covering the basics of this area. An expert on a particular field has higher expectations from his search iterations. The knowledge of searchers in particular areas depends to a large extent on various factors and is considered to be query-dependent.

The user's knowledge can be described as a relevance threshold t_i. It represents how much relevant a document has to be to attract the user. The relevance of each document is derived by measuring the cosine similarity between the terms in the document and term set in the user's profile. Since a user's profile represents a snapshot of his current search needs, the similarity between this and the document term set exhibits how relevant the current document is to his search interest.

7.3.4 Creating a Searcher Simulation Model

User patience, knowledge and overall search experience will be incorporated in our simulation model. In our simulation, each search iteration must take into account these factors and produce the results accordingly.

Based on these principles, a simulated user can be thought as a tuple of values (ti, pi, ei), where ti the threshold for this user, pi is the patience for this user and ei is the user's experience value. Thus, by defining a new tuple, we introduce a new searcher in our environment.

Throughout this experiment, I simulate a user by using a searcher stereotype, browsing and interacting with search results. All interactions within each simulation are with the set of retrieved documents for a query. During this simulation study, a large set of searchers have been defined in order to examine different behaviors in a common environment. For instance, a searcher with high threshold and knowledge, but low patience, can be used to simulate an expert in a field. Nonetheless, the searcher stereotypes definitions presented below cannot be regarded as accurate simulations of real people. The aim is to gain further insight to the recommendation and profiling process by employing searcher simulations. The following list associates a range of searcher simulation tuples with an *approximate searcher stereotype*:

1. (t = 0, p = 100, e = 0): Beginner searcher with high patience but no previous knowledge
2. (t = 10, p = 10, e = 10): Average searcher with high knowledge of the subject
3. (t = 0, p = 100, e = 20): Expert searcher with high patience but no previous knowledge
4. (t = 5, p = 30, e = 30): Expert searcher with high patience and some previous knowledge

5. (t = 8, p = 25, e = 20): Expert searcher with high patience and some previous knowledge

6. (t = 15, p = 10, e = 5): Above average searcher with very high knowledge of the subject

7.4 Simulation Components

Due to their degree of sophistication, modern adaptive retrieval algorithms incorporate a large number of parameters that support the functionality of such context-aware applications. The underlying weighting scheme, adaptive window size as well as similarity matching thresholds account for the performance of profiling techniques and algorithms. To our best knowledge, there has been no previous attempt to optimize the performance of adaptive information retrieval systems by fine-tuning individual components and parameters.

As part of our simulation methodology, I benchmark various system components to study thoroughly the effects of individual settings and derive an optimal configuration. I refer to these parameters as *simulation components*. Optimal settings for each simulated component are our ed by fixing the remaining parameters and performing a full evaluation run. The simulation components in Colombo are follows:

1. Term weighting schemes: Nine term weighting schemes have been incorporated in Colombo (Boolean, Voting Boolean, TF, Voting TF, TF-IDF, Voting TF-IDF, WPQ, Voting WPQ, Jeffrey's conditioning). Each of these models have been included in the simulation experiments.

2. Similarity matching thresholds: Colombo's profiling algorithm is based on a two-level filtering module to represent user interests as group of topics. Common interests are merged together, less similar topics are grouped in the same cluster, while a new cluster is created for distant topics. This hierarchy of user interests is created through a two-level K-nearest neighbour algorithm based on two thresholds (interest/clustering). I argue that varying these thresholds can return different results for each setting. This aspect has also been integrated in our simulation study in order to discover where each threshold configuration is more appropriate. I have included 8 pairs of threshold settings:

(a) Clustering threshold = 0.15, Interest matching threshold = 0.25

(b) Clustering threshold = 0.15, Interest matching threshold = 0.35

(c) Clustering threshold = 0.25, Interest matching threshold = 0.25

(d) Clustering threshold = 0.25, Interest matching threshold = 0.35

(e) Clustering threshold = 0.25, Interest matching threshold = 0.40

(f) Clustering threshold = 0.35, Interest matching threshold = 0.35

(g) Clustering threshold = 0.35, Interest matching threshold = 0.40

(h) Clustering threshold = 0.35, Interest matching threshold = 0.50

3. Simulated searcher: As detailed along section 7.3.4, a range of searcher stereotypes have been defined and employed throughout this experiment to simulate the effect of real users searching the system. Searcher stereotypes allow simulated experiments to partially include and analyze user aspects such as patience and experience. A total number of 6 searcher stereotypes have been defined during this experiment ranging from beginner to experiences searcher models (Section 7.3.4).

7.5 Experimental procedure

We argue that searcher simulation based on the above principles can provide indications about the performance of different system components. As mentioned in an earlier section, I have conducted such offline experiments based on the theory of searcher simulations in order to tune our system appropriately. Conclusive results concerning the effectiveness of algorithms incorporated in Colombus have been drawn by employing a user-centered evaluation methodology. Chapter 8 presents the experiments conducted as part of this approach in detail.

7.6 Evaluation Measures

This section covers the evaluation measures incorporated as part of our simulation framework. The primary measure used throughout this study is an *11-pt precision for document recommendations* per

search iteration averaged across all experimental runs. I measure the performance of the recommendation component on a per-iteration basis. Since the advertised function is document recommendations, I assume that searchers are looking to maximize this precision value through their search episodes.

Also, a set of correlation-based measures has been employed to benchmark the profiling scheme. The correlation between relevant documents of each query and the extracted term set (used to augment the user's profile), as well as the terms in the user's profile, are also taken into account. In particular, I calculate the *cosine distance* between the terms in relevant documents and (a) terms extracted from search results after relevance feedback, (b) terms in user's profile. These two metrics can provide a performance indication with regards to the adaptive profiling model.

Although several studies have used profile accuracy as a measure to evaluate profiling algorithms, the trend of information capturing has still to be shown. Profile accuracy as an evaluation measure for adaptive systems is restrictive as researchers cannot see how profiles matured during their lifetime. Instead of calculating the profile accuracy at the end of each run, I argue that the evolution trend of profiles throughout the search iterations is more appropriate. I propose a novel evaluation measure to capture this particular effect. Profile Evolution Trend (PET) measure is calculated at the end of each search iteration by comparing the actual number of interests in a profile against the expected number of interests. For instance, after two distinct search episodes about "*digital cameras*" and "*programming techniques*" the expected number of interests in the user's profile is two. By setting the expected number of interests as the optimal baseline, I examine the correlation of the actual profile to this baseline. PET has been employed during our simulated study and has served as the main metric for measuring profile accuracy.

These measures have been used throughout the simulation scenarios. However, depending on the scenario methodology, their degree of importance varies. Section 7.7 covers the approach to each scenario in more detail.

7.7 Simulation Scenarios

In this section I describe the scenarios that have been included in our simulation study. Each of these scenarios benchmarks a different aspect of Colombo, focussing on the profiling and recommendation components. On the other hand, each simulation run is associated with a TREC topic/query and includes a series of search iterations which approximates the users' quest for relevant information. Relevance feedback is captured after each search by simulating random clicks on documents representations. Subsequently, extracted terms represent the future query, while they are also used to augment the user's profile. At the end of each search iteration evaluation measures are recorded for further examination. Also, Wilcoxon test is employed where applicable to measure the statistical significance of results.

7.7.1 Scenario 1

The first experiment attempts to capture the retrieval effectiveness of our information assistant in recommending relevant documents with respect to the users' needs. In this aspect, I employ the evaluation measures described in section 7.6 throughout a total of 50 simulation runs. Our objective is to study the effectiveness of term weighting schemes combined with a range of threshold settings and analyse the results with respect to the system's performance in terms of three evaluation measures: (i) recommendation precision, (ii) correlation between profile terms and relevant documents and (iii) correlation between query terms and relevant documents.

Only one topic is simulated in this scenario and the underlying objective for adaptive models is to optimize document recommendations. The simulation procedure for this particular experiment is as follows:

1. Create a simulated user.
2. Get query from TREC topics list
3. Issue query and fetch results
4. Simulate relevance feedback.

(a) For each document in search results:

i. If document above threshold then visit

ii. Continue till no more results or patience value is less/equal to 0

(b) Introduce noise in search results based on user's experience

5. Extract terms from relevant documents and amend profile.

6. Calculate and record the correlation between generated term set and relevant document set as well as between users profile and relevant document set

7. Simulate document recommendation

8. Calculate and record evaluation measures

9. Extracted terms from step 5 is new search query

10. Start from step 3 and repeat for 10 iterations

7.7.2 Scenario 2

The previous experiment mainly assesses the performance of the system in terms of recommendation precision. However, the behaviour of Colombus in gradual changes of relevance (concept drifts) also needs to be evaluated. Triggered by interaction with new information, an existing information need may change, drifting to a similar concept (Belkin 1997). In this scenario, I experiment with the adaptation mechanism of our system by gradually introducing a slow, steady change from an existing information need to a highly similar topic. Since the two topics are close together, I expect that the profile learning scheme will detect the slight change in relevance and update the existing interest, instead of creating a separate one in the user's profile.

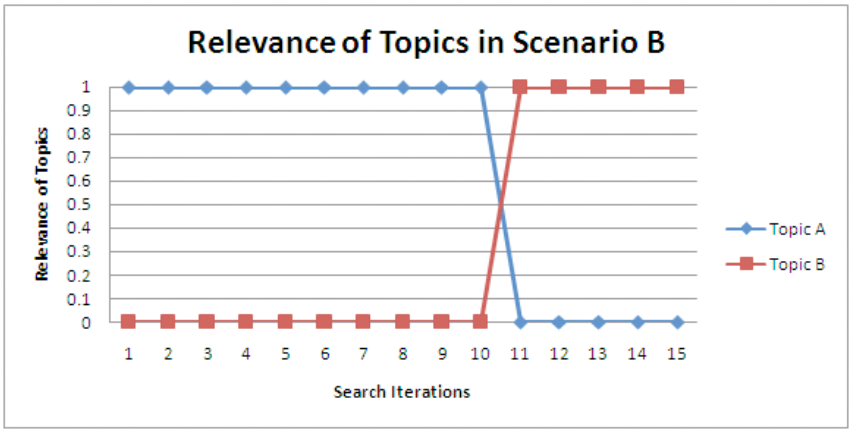

Figure 7.1: Relevance of topics in the second scenario

The similarity between two topics is defined by the common range of associated relevant documents in TREC judgments. A pair of topics A and B with 5 relevant documents in common are considered more similar than a pair of topics C and D with no relevant judgments in common. Setting a similarity threshold of 25%, I discovered two pairs of topics to examine in this scenario. Each experimental run consists of a total of 20 search iterations. The experimental procedure is detailed in a step-by-step list below:

1. Create a simulated user.
2. Get query A from TREC topics list
3. Get query B from TREC topics list, such that Q_A is 25% similar to Q_B
4. Issue queries and fetch results
5. Simulate relevance feedback as detailed in figure 7.1
   (a) For each document in search results:
      i. If document above threshold then visit
      ii. Continue till no more results or patience value is less/equal to 0
   (b) Introduce noise in search results based on user's experience

6. Extract terms from relevant documents and amend profile.
7. Calculate and record the correlation between generated term set and relevant document set as well as between users profile and relevant document set
8. Calculate the Profile Evolution Trend measure (PET)
9. Simulate document recommendation
10. Calculate and record evaluation measures
11. Extracted terms from step 4 is new search query
12. Start from step 3 and repeat for 20 iterations

7.7.3 Scenario 3

Measuring the effectiveness of a recommendation service in terms of recommendation precision alone has been found to be insufficient (Herlocker, Konstan, Terveen & Riedl 2004). An accurate representation of information needs, fast learning rate, as well as user satisfaction and recommendation novelty also need to be accommodated in such an information filtering system.

The third scenario studies the effect of a temporal topic in the profiling algorithm. I simulate an abrupt concept shift and examine the effect in the profile structure. In a multi-facet profile environment, like Colombus, an unexpected short-term information need must be represented as a distinct interest in the user's profile. Otherwise, an existing interest will absorb the information gathered and clutter the profile. In the current test case, I assume the presence of a user interest, while a temporal topic is introduced in the system. By simulating 10 searches, spread out over 30 days, on a particular topic, a relatively stable, information need is developed and included in the user's profile. Then, an abrupt concept shift is simulated, which should lead to another interest being inserted in the user's profile.

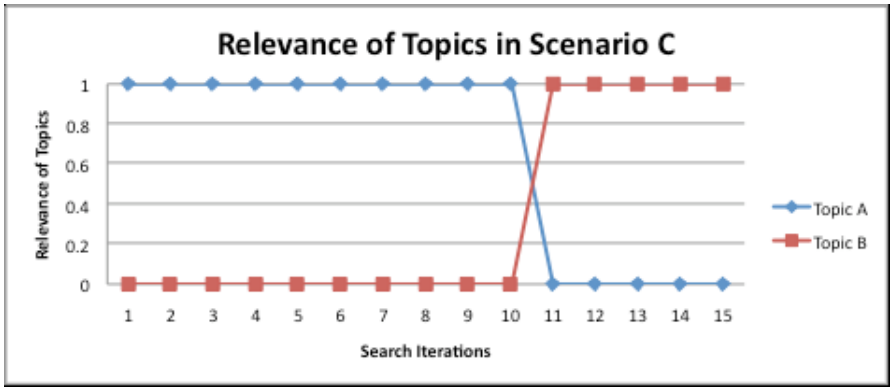

Figure 7.2: Relevance of topics in the second scenario

The results are analyzed in terms of the profile evolution, i.e. how the number of interests have oscillated over time. In the event where the temporal topic has been successfully detected, there is no need to re-evaluate the precision of recommendations, as it has already been currently discovered as part of the first scenario.

When the above process is completed, the appearance of a temporal topic is simulated, based on the existing simulated searcher. Searches on this particular topic are assumed to be issued on the same day, resembling the need for information when a temporal topic emerges. Five queries are issued, as part of this process, following the same procedure described above.

1. Create a simulated user.
2. Get query A from TREC topics list
3. Get query B from TREC topics list, such that Q_A is completely dissimilar to Q_B
4. Issue queries and fetch results
5. Simulate relevance feedback as detailed in figure 7.2
   (a) For each document in search results:
      i. If document above threshold then visit
      ii. Continue till no more results or patience value is less/equal to 0

(b) Introduce noise in search results based on user's experience

6. Extract terms from relevant documents and amend profile.
7. Calculate and record the correlation between generated term set and relevant document set as well as between users profile and relevant document set
8. Calculate the Profile Evolution Trend measure (PET)
9. Simulate document recommendation
10. Calculate and record evaluation measures
11. Extracted terms from step 4 is new search query
12. Start from step 3 and repeat for 15 iterations

7.8 Discussion

In the absence of a proper methodology for evaluating adaptive recommendation algorithms and systems, I introduced an evaluation methodology based on searcher simulations. Searchers actions were simulated in terms of page visits to generate implicit relevance feedback. Different scenarios have been simulated in order to gain more insight into the performance of a recommendation system. Through this evaluation methodology, I attempt to simulate a user-centered study, while at the same time avoiding the problems with real-life experimental setting. This has allowed us to benchmark our algorithms and hence fine tune various parameters.

This methodology is not a replacement for real-user study but a preliminary attempt to gather performance issues. A potential drawback of such methodology is that it won't consider the intentionality in interaction. A real searcher will view a series of information objects in a rational way, depending on their information need (Jansen & Pooch 2001).

Another drawback of simulated experiments in general is that they exclude user interface components from the experimental procedure. Cognitive aspects of the system are ignored, yet recent studies have

highlighted the importance of user interface designs in information filtering applications (Liu, Wong & Hui 2003).

7.9 Simulation Statistics

After carrying our the simulation-based evaluation, it is interesting to analyze and present a range of statistics to demonstrate the scale of this study. Three search scenarios were carried out through this large-scale simulation study and a range of application settings were tested to calibrate the system effectively. As detailed in section 7.4, a total number of 6 searcher stereotypes were simulated for each experimental scenario, while 8 threshold configuration were also examined. Therefore, *each experimental scenario was carried out* ***48 times*** in order to collect the required data.

Also, depending on the underlying objective, the number of experimental runs varied for each search scenario. More specifically, a total of 50 experimental runs was carried out for the first and third scenario, while only a couple of runs was executed for the second scenario due to its lengthy setup conditions. Similarly, the number of search iterations for each experimental scenario also varied significantly. A total number of 10, 15 and 20 search iterations were simulated for the first, second and third scenario respectively. As a result, *we simulated* ***5280 experimental runs*** throughout the entire duration of this simulation study, while *a total number of* ***69600 search iterations*** *were issued.*

Chapter 8

Simulation Result Analysis

This study was conducted to optimize the performance and explore different system settings towards adaptive profiling of user needs for recommender systems. Although I have focused on the performance of term weighting schemes, I also present the results of our study in the grounds of similarity matching thresholds and searcher models. The evaluation results collected by experimenting with different term weighting schemes are considered of higher importance than the remaining simulation components. Therefore, I first discover and establish the upper set of weighting schemes in terms of performance.

The rest of this chapter is structured as follows. The experimental results of scenarios 1, 2 and 3 are presented in sections 8.1, 8.3 and 8.2 respectively. Then, section 7.8 discusses the results of our simulation study and sets an optimal configuration for use in subsequent user experiments.

8.1 Scenario 1

8.1.1 Evaluating Term Weighting Schemes

In this scenario, I measured the retrieval effectiveness of document recommendations across different term selection strategies and searcher stereotypes (i.e. user models). Initially, I assume the presence of

a beginner searcher stereotype with tuple (t=0, p=100, e=0) and profiling thresholds equal to 0.25 and 0.40 for cluster and document threshold respectively. Figure 8.1 plots 11-pt precision values grouped by search iteration and averaged across 50 experimental runs. The TF-IDF and voting TF outperformed the rest of the term weighting schemes. All schemes balance to an almost fixed accuracy figure after approximately 6 search iterations. As more information (i.e. terms) are extracted and stored in the user's profile, the influence of information gathered during new search iterations decreases. The lower band of figure 1 includes WPQ-based, as well as Jeffrey's conditioning and Voting TF-IDF, schemes, which resulted in a rather poor performance.

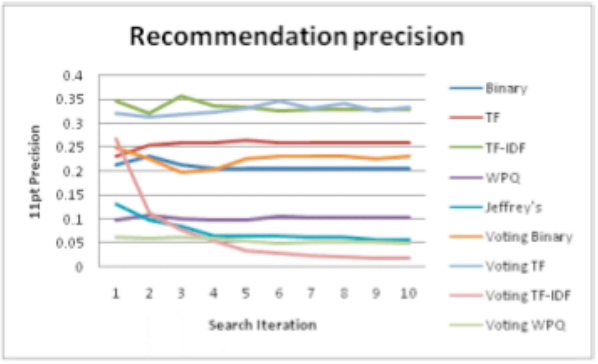

Figure 8.1: Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the first scenario

Table 8.1 illustrates marginal differences in recommendation precision across all iterations and is closely connected to the rate of learning for each term weighting scheme. The results for each search iteration are distributed into two columns; the left presents the recommendation precision at that iteration, while the right one shows the *marginal difference from the previous point* in italic. The best performer, the one with the highest recommendation precision, for each search iteration is shown in bold. The rate at which information gathered through new search episodes is taken advantage by the profiling module is analogous to the rate of learning for each weighting scheme. As table 8.1 illustrates, the voting TF scheme shows a steady increase in precision as more information is accumulated in the profile vectors,

while TF-IDF shows less consistency in performance. The remaining weighting schemes follow a similar performance deviation trend to TF-IDF.

	Search Iteration									
Weighting Scheme	**1**		**2**		**4**		**8**		**10**	
Binary	0.21	-	0.23	*0.02*	0.20	*-0.03*	0.20	*0*	0.20	*0*
TF	0.23	-	0.25	*0*	0.25	*0.01*	0.26	*0*	0.26	*0*
TF-IDF	**0.34**	-	**0.32**	*-0.02*	**0.33**	*0.01*	0.32	*-0.01*	0.32	*0*
WPQ	0.09	-	0.10	*0.01*	0.09	*-0.01*	0.10	*0.01*	0.10	*0*
Jeffrey's	0.13	-	0.09	*-0.04*	0.06	*-0.03*	0.06	*0*	0.05	*-0.01*
Voting Binary	0.25	-	0.22	*-0.03*	0.20	*-0.02*	0.23	*0.03*	0.23	*0*
Voting TF	0.32	-	0.31	*-0.01*	0.32	*0.01*	**0.33**	*0.01*	**0.33**	*0*
Voting TF-IDF	0.26	-	0.11	*-0.15*	0.05	*-0.06*	0.02	*-0.03*	0.01	*-0.01*

Table 8.1: Marginal difference in recommendation precision between search our tions for each term weighting scheme

Although this scenario largely focuses on the recommendation component, the effect of each search iteration on the user's profile is also studied. The cosine correlation-based measures also indicate each scheme's rate of learning. Figure 8.2 presents the correlation between the profile terms and the relevant documents, while figure 8.3 shows the correlation between relevant documents and query terms. Schemes that show a steep correlation increase are considered to create and amend profiles more effectively. Although the voting TF and binary weighting schemes start at a high degree of correlation in both graphs, they remained relatively constant throughout the search episodes. On the other hand, as figure 8.2 demonstrates, TF-IDF maintained a steady our and a particularly steep rate of learning towards the end.

The previous set of results has assumed an optimal searcher stereotype, where many documents are visited in a noise-free search environment. This section studies the effect of noisy relevance judgments achieved by simulating a less experienced searcher stereotype with tuple (t=0, p=100, e=20). Figure 8.4 displays the average precision for document recommendations. The results follow a similar trend to figure 8.1,

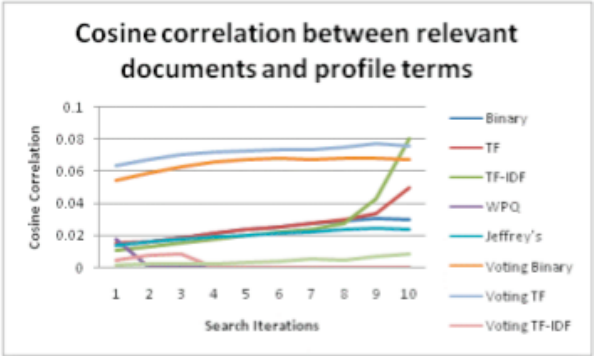

Figure 8.2: Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the first scenario

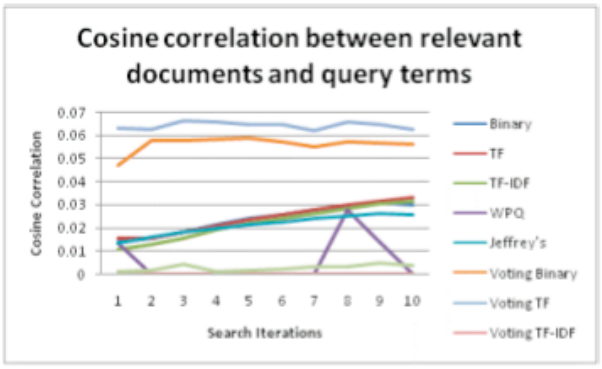

Figure 8.3: Correlation between the query terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the first scenario

presented in the earlier section. Again, TF-IDF and voting TF weighting schemes produce the best results, although the difference in performance between weighting schemes is reduced. Jeffrey's conditioning and WPQ based schemes perform poorly, because their recommendation performance deteriorates throughout the experiment.

As far as the observed learning rate is concerned, there is a similar trend to the previous correlation graphs (figures 8.5, 8.6). In the profile correlation graph, where the cumulative profiling performance is observed, WPQ has outperformed the rest of the term weighting schemes, but considering its performance

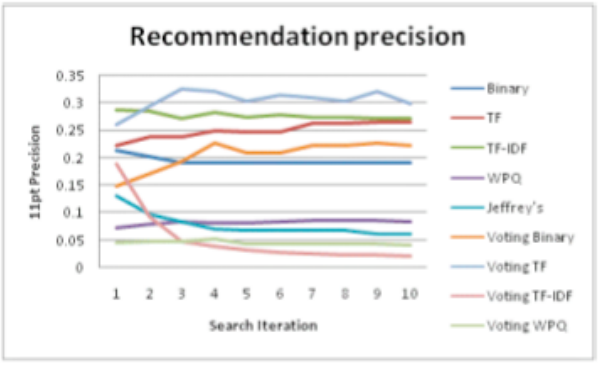

Figure 8.4: Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the first scenario

so far, the results are rather measure and user dependent. This becomes apparent in figure 8.6 where the WPQ weighting scheme showed inconsistent performance surges. A large group of weighting models have illustrated a homogeneous monotonous rate of learning with an insignificant our in profile/query correlation. In essence, these particular models failed to extract useful terms relevant to the user's information need and, ultimately, augment his profile effectively. On the other hand, TF-IDF and TF (voting and non-voting) schemes demonstrate a slow, but consistent correlation increase (figure 8.5) throughout all search iterations and their performance is significantly superior than the remaining weighting schemes (Wilcoxon test p=0.003). Taking into account the noisy search environment and the results of these schemes throughout this experiment, we can project an indication with respect of their reliability in document recommendations.

The results presented in this section focused on the performance of Colombo assuming the presence of two simulated users with a high degree of patience (*p=100*) and variable experience/noise (*e=0*, *e=20*). However, it is rather uncommon for searchers to browse through such a large set of documents to discover relevant information. Recent studies have highlighted that the approximately 80% of web searchers view no more than two result pages, which translates to 10-20 documents (Jansen & Spink 2005, Meng 2006). The remainder of this section illustrates the results of this experiment when simulating the presence of a

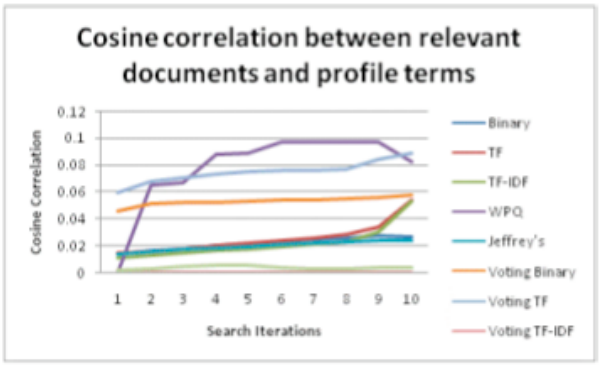

Figure 8.5: Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the first scenario

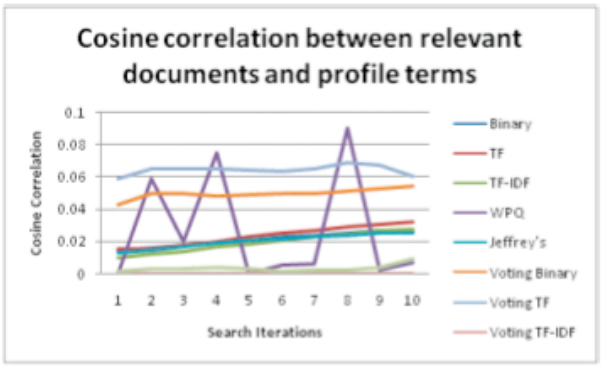

Figure 8.6: Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the first scenario

more realistic searcher stereotype (t=10, p=10, e=10) and presents a cumulative summary for all searcher models. This particular simulated searcher exhibits a *selective document visiting* behaviour, since only results above a certain threshold are visited. At the same time, a certain degree of noise (10%) is injected to the relevance judgment set in order to simulate a less optimal search environment.

Also, I present the results of average recommendation precision for a searcher stereotype with tuple (t=10, p=10, e=10). Figure 8.7 demonstrates a homogeneous decrease in precision values compared to previous searcher models. However, TF-IDF and Voting TF still remain the top performing weighting schemes. A similar trend is observed for the lower bands of figure 8.8, where Voting TF-IDF and WPQ schemes result in performance degradation throughout all search iterations.

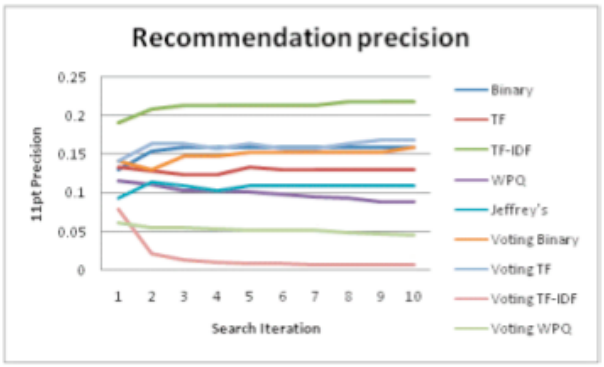

Figure 8.7: Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=10, p=10, e=10) in the first scenario

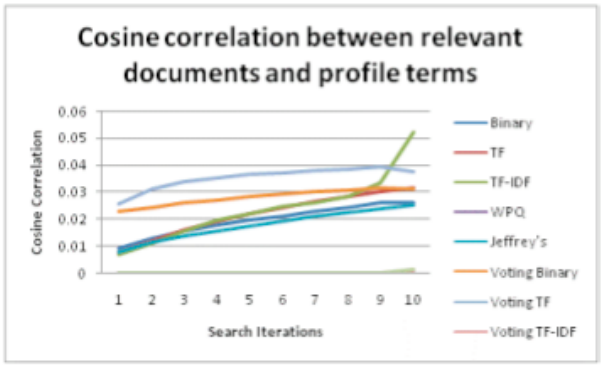

Figure 8.8: Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=10, p=10, e=10) in the first scenario

In addition, figure 8.8 examines the rate of learning for the particular searcher stereotype through the correlation of profiles and relevant documents. We can observe an analogous trend to our correlation figures presented earlier in this section, while the performance drop constitutes an expected effect of the noisy environment and user selective behaviour. However, interestingly enough, the performance gap between term weighting schemes has decreased. A close examination of the profile correlation figures presented so far demonstrate that it is actually the voting TF and binary schemes which deteriorate, while the rest remain relatively unchanged to different searchers.

This particular effect is further examined in table 8.2, which demonstrate the marginal differences in profile correlation across all search iterations for three simulated searchers. The marginal difference is shown in a separate column next to each iteration and indicates the correlation difference between the current and the previous iteration. Each row summarizes the results for a particular searcher stereotype. Although the presentation of these tables implies the examination of the marginal difference on a per-iteration basis, our main focus is to study the difference in performance for each searcher stereotype and weighting scheme.

	Iterations									
Searcher Stereotype	**1**		**2**		**4**		**8**		**10**	
	Voting Binary									
(t=0, p=100, e=0)	0.053	-	0.058	*0.05*	0.065	*0.07*	0.068	*0.03*	0.067	*-0.01*
(t=0, p=100, e=20)	0.045	-	0.050	*0.05*	0.052	*0.02*	0.054	*0.02*	0.057	*0.03*
(t=10, p=10, e=10)	0.022	-	0.024	*0.02*	0.027	*0.03*	0.030	*0.03*	0.030	*0.00*
	Voting TF									
(t=0, p=100, e=0)	0.063	-	0.067	*0.04*	0.071	*0.04*	0.074	*0.03*	0.075	*0.01*
(t=0, p=100, e=20)	0.059	-	0.067	*0.08*	0.073	*0.06*	0.077	*0.04*	0.088	*0.11*
(t=10, p=10, e=10)	0.025	-	0.031	*0.06*	0.035	*0.04*	0.038	*0.03*	0.037	*-0.01*
	Iterations									
Searcher Stereotype	**1**		**2**		**4**		**8**		**10**	
	Binary									
(t=0, p=100, e=0)	0.013	-	0.015	*0.02*	0.021	*0.06*	0.028	*0.07*	0.029	*0.01*
(t=0, p=100, e=20)	0.014	-	0.016	*0.02*	0.020	*0.04*	0.026	*0.06*	0.027	*0.01*
(t=10, p=10, e=10)	0.009	-	0.009	*0.00*	0.017	*0.08*	0.024	*0.07*	0.026	*0.02*
	TF									
(t=0, p=100, e=0)	0.013	-	0.015	*0.02*	0.018	*0.03*	0.023	*0.05*	0.023	*0.00*
(t=0, p=100, e=20)	0.013	-	0.015	*0.02*	0.018	*0.03*	0.022	*0.04*	0.023	*0.01*
(t=10, p=10, e=10)	0.007	-	0.011	*0.04*	0.015	*0.04*	0.022	*0.07*	0.025	*0.03*

Table 8.2: Marginal differences in profile correlation across all search iterations for three simulated searchers for each term weighting scheme

Although this experiment has mostly examined the effect of different term weighting schemes in the document recommendation and user profiling process, information retrieval systems are used by searchers with a diverse backgrounds. Due to the scale of this simulation study we are not able to present and study

all the results for all possible permutations of configurations and searcher stereotypes. However, to gain further insight on the effect of different user models, I have summarized and present the precision values in figure 8.9. The precision values have been averaged for all searcher stereotypes and are presented on a per-iteration basis for each term weighting scheme.

As figure 8.9 demonstrates, TF-IDF and voting TF weighting schemes have outperformed the remaining models. Based on the results of previous figures in this scenario, we can conclude that these models are more robust and reliable, while the quality of recommendations has always been kept to high standards. (Wilcoxon test p=0.003). From the remaining models, Jeffrey's conditioning and voting TF-IDF have failed our expectations, as their performance degrades along search iterations.

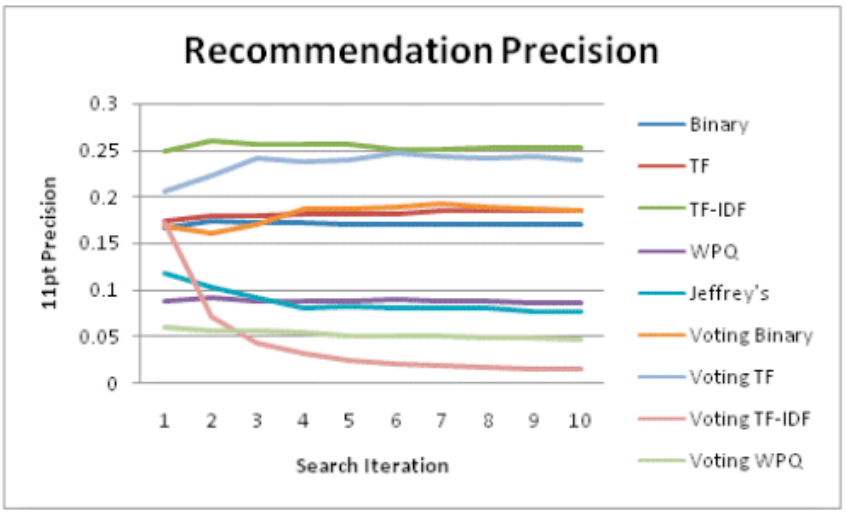

Figure 8.9: Average recommendation precision of each term weighting scheme aggregated for all searcher stereotypes in the first scenario

In this section the underlying document recommendation component was tested using an 11pt precision measure, while the correlation-based metrics have benchmarked the profiling module in terms of rate of learning. Only the results of different weighting schemes have been presented an analyzed, while a series of searcher stereotypes have been simulated. We observed that TF-IDF and voting TF perform equally

well and the results were independent of the searcher stereotype employed. As far as the profiling process is concerned, the results indicate that voting TF and binary schemes successfully track and accommodate information in user profiles. In the next section, I evaluate the document recommendation component across a range of profiling configurations in regards to clustering thresholds.

8.1.2 Evaluating Clustering/Document Threshold Configurations

Recommendation systems are based on a learning mechanisms to cater for the needs of individual searchers. Colombo has adopted a two-stage nearest-neighbour approach to user profiling, where a pair of thresholds accommodate new information in the existing profile. As a result, these threshold values heavily influence the performance of the underlying profiling algorithm and recommendation component. In this section I evaluate a range of threshold values and examine their influence on document recommendations.

After the term weighting schemes has been benchmarked as part of section 8.1.1, it was shown that TF-IDF and voting TF result in higher quality document recommendations across different search environments and user stereotypes. Since no definite conclusion could be drawn at this stage as to which weighting model is more appropriate, I present the results of this scenario for each of these two schemes. To maintain simplicity throughout the result figures, only the top performing weighting schemes, TF-IDF and voting TF, have employed at this stage.

As in the previous subsection of this scenario, we start our analysis by simulating the user behaviour of a searcher stereotype with tuple (t=0, p=100, e=0). Figure 8.10 illustrates the results of recommendation precision for a series of different threshold pairs. As it is shown, both weighting models perform closely, whereas TF-IDF results have been identical throughout the search iterations. Although the combination of voting TF scheme and a threshold pair of CT=0.25 and DT=0.35 appears to outperform the remaining configurations, it is shown that its performance has not been robust through the duration of this experiment. On the other hand, the same term weighting scheme based on more aggressive threshold settings, such as CT=0.35 and DT=0.50, demonstrates an increasing performance trend through the majority of search iterations.

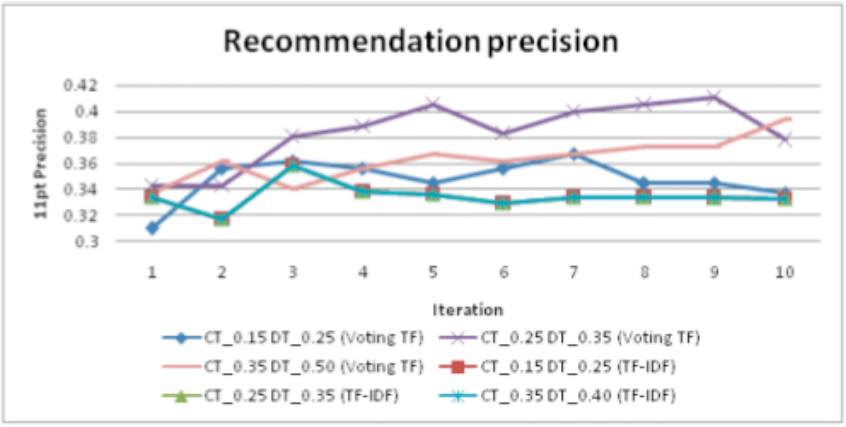

Figure 8.10: Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario

Furthermore, after analyzing the results obtained from the correlation-based metrics, I observed that all threshold configurations perform very closely. This particular effect is demonstrated in figure 8.11, where I present the correlation between relevant documents and profile terms. In fact, a similar trend has been observed irrespective of the underlying searcher stereotype employed. Therefore, correlation-related figures have been omitted from the remainder of this section.

Interestingly enough the result trends change as a different searcher stereotype is simulated (t=0, p=100, e=20). The simulation of such a user aims to examine the effect of noise in the overall retrieval performance. Figure 8.12 presents the results for this particular configuration. TF-IDF is shown to outperform the other weighting schemes/threshold settings, while a similar performance is observed for the remaining models. This comes in opposition to the results obtained for a different searcher stereotype, presented in figure 8.10

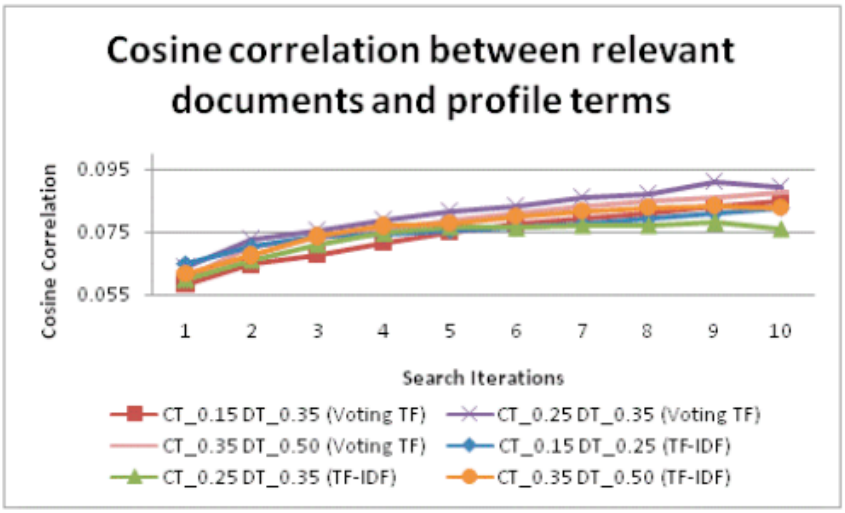

Figure 8.11: Correlation between the profile terms and the relevant documents of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario

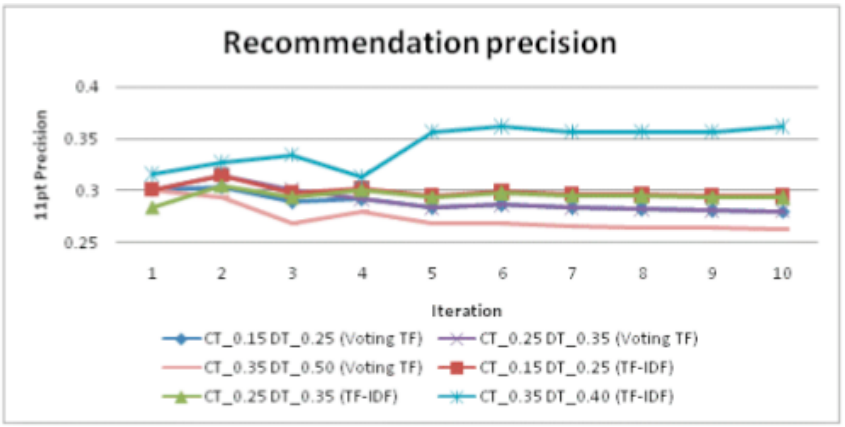

Figure 8.12: Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the second scenario

Finally, the average performance for all searcher stereotypes in terms of recommendation precision is summarized in table 8.3. Although two different weighting schemes are combined with a range of threshold settings, no particular configuration returns significantly better results. Actually, no specific weighting scheme/threshold setting produces better results throughout all iterations. However, we can observe that higher cluster/document threshold pairs seem to perform slightly better than their lower counter parties.

For instance, the third entry in the table (CT=0.35, DT=0.50, Voting TF) produces better results than the first entry in the table (CT=0.15, DT=0.25, Voting TF) (Wilcoxon test p=0.003).

	Search Iteration									
Configuration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	10
CT_0.15 DT_0.25 (Voting TF)	0.220	0.237	0.241	0.242	0.247	0.240	0.244	0.239	0.240	0.236
CT_0.25 DT_0.35 (Voting TF)	0.230	0.243	0.250	0.245	0.249	0.245	0.250	0.247	0.247	0.244
CT_0.35 DT_0.50 (Voting TF)	0.240	0.260	0.247	0.253	0.257	0.253	0.253	0.254	0.251	0.255
CT_0.15 DT_0.25 (TF-IDF)	0.237	0.251	0.248	0.250	0.250	0.248	0.249	0.250	0.251	0.251
CT_0.25 DT_0.35 (TF-IDF)	0.231	0.241	0.244	0.246	0.248	0.243	0.246	0.246	0.246	0.246
CT_0.35 DT_0.40 (TF-IDF)	0.244	0.255	0.259	0.259	0.269	0.266	0.328	0.267	0.268	0.268

Table 8.3: Average performance of threshold settings and term weighting schemes on a per-iteration basis

After analyzing all results obtained through this experiment, it was observed that all searchers simulations follow the same trend from the correlation-based metrics, we observed that all threshold configurations perform very closely. In fact, a similar trend has been observed irrespective of the underlying searcher stereotype employed. Therefore, correlation-related figures have been omitted from the remainder of this section. Based on the results presented in this section, no definite conclusions could be drawn as to which term weighting scheme produces better results. Both voting TF and TF-IDF outperformed each other in different simulation runs, while their performance aggregated for all searcher stereotypes was very similar. In the following sections, I will further examine the profiling process by simulating two concept change scenarios.

8.2 Scenario 2

In this second scenario, I simulated a ”concept drift” in the user's search preferences to analyse the adaptivity of document recommendations in Colombo. In essence, such drifts are *facet transitions* that often occur in marginal situations, such as when new search needs appear. In effect, as users collect more information during their search episodes, their perspective of their search interest is likely to change. Commonly, users start a new information quest from a general point of view and slowly drift to more specific facets of the topic, however even persistent search needs may slightly change after long-term information gathering. As detailed in section 7.7.2, the results have been gathered by simulating a constant, gradual drift from one interest to another, while experimenting with several term weighting schemes, threshold configurations and searcher stereotypes. The performance of Colombo throughout this experiment has been measured in recommendation precision and our own PET metric. The remainder of this section is divided into two parts. Section 8.2.1 examines the effect of different weighting schemes in the event of a concept drift, while section 8.2.2 presents the results obtained after attempting a range of threshold settings.

8.2.1 Evaluating Term Weighting Schemes

For this first part of this scenario I assume constant threshold settings of 0.25 and 0.35 for cluster and document thresholds respectively, while simulating a searcher stereotype with tuple (t=0, p=100, e=0). Figure 8.13 presents the performance of each term weighting scheme in terms of average precision of document recommendations. A sudden drop in average precision can be noticed on the 10th search iteration, when the drift to a similar topic starts to occur. Since Jeffrey's conditioning and voting TF-IDF schemes perform equally poor to the first scenario, our observations focus on the remaining models. TF-IDF appears to be the only term weighting scheme that accommodates concept drifts in an acceptable way. Further analysis showed that the selection of default threshold settings proved to be the reason behind such inconsistencies. This particular issue is covered in more detail along section 8.2.2.

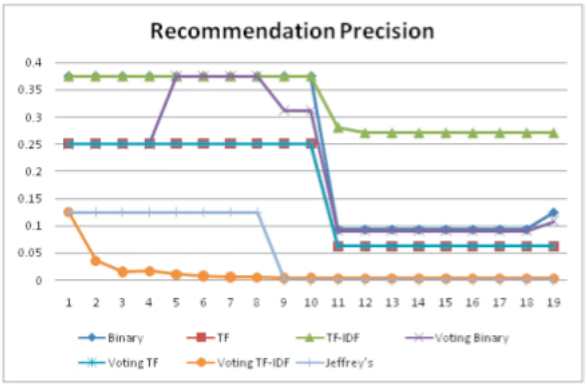

Figure 8.13: Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario

Also, data gathered through the PET measure are suitable to discover weaknesses in the profiling process. Figure 8.14 presents the PET results, where the baseline represents the optimal profile evolution. Since this scenario only simulates a gradual concept drift between common facets of the same topic, the user's profile should have only one interest throughout search iterations. We can see that the results for TF and voting TF weighting schemes coincide, while their average performance is close to optimal. Although two common aspects of a single topic were used in this experiment, it can be observed that several schemes , such as TF-IDF and Jeffrey's conditioning, recognized the multi-facet nature of user interests. As a result, the introduction of a new facet of an already explored topic triggers the creation of a new interest in the profile vector. However, the rate of drift should be analogous to the rate profile change, hence the start of a slow drift at the mid-point of search iterations should not cause an immediate change to the underlying profile structure. A very similar set of results have been observed for a simulated searcher with tuple (t=5, p=30, e=30) and are shown in figure 8.15 for reference purposes.

Adapting the same analysis methodology followed in earlier experiments, I also present the results for a different searcher stereotype (t=0, p=100, e=20) in figures 8.16 and 8.17. Figure 8.16 summarizes average recommendation precision, while figure 8.17 focuses on the profile evolution process. Further analysis of figure 8.16 strengthens earlier observations that the TF-IDF weighting scheme is less affected

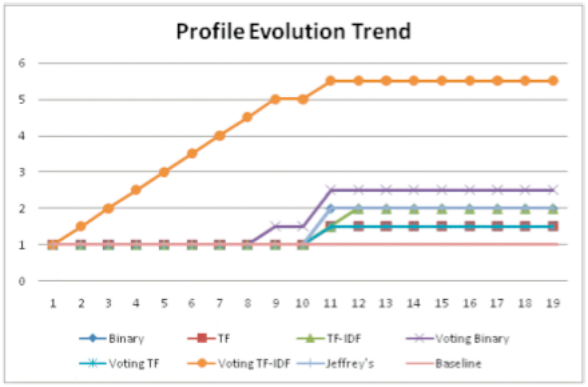

Figure 8.14: The profile evolution trend (PET) results of each term weighting scheme averaged for searcher stereotype with tuple (t=0, p=100, e=0) in the second scenario

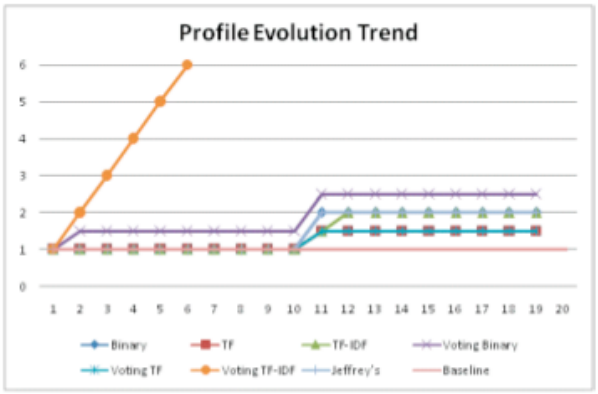

Figure 8.15: The profile evolution trend (PET) results of each term weighting scheme averaged for searcher stereotype with tuple (t=5, p=30, e=30) in the second scenario

by the concept change compared to the remaining models. At the same time, voting TF-IDF continues to perform poorly, while the performance of remaining schemes falls dramatically after the concept change.

Figure 8.17 studies the evolution of profiles over time, focussing on the marginal point of the concept change. The results are very similar to those obtained for different searcher stereotypes and presented earlier in this section. However, there appears to be a performance drop from the voting binary scheme as a result of the presence of noise in the search environment. The remaining models illustrate a similar

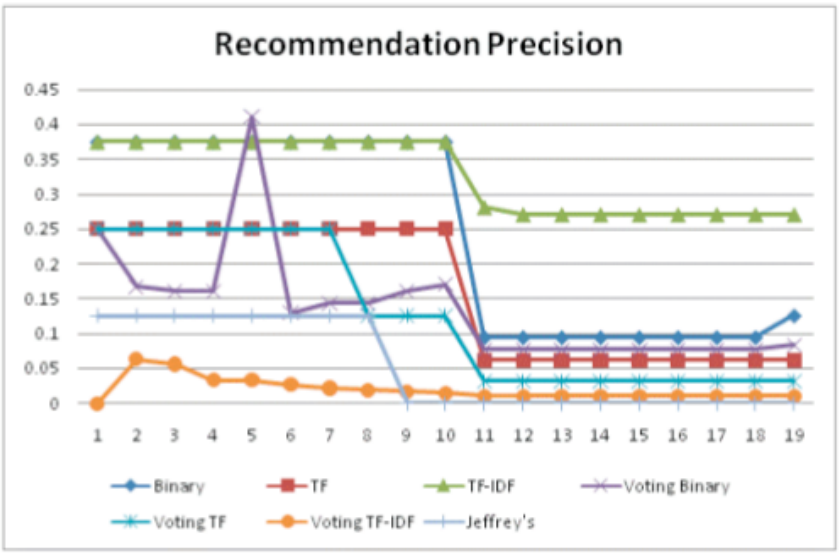

Figure 8.16: Average recommendation precision of each term weighting scheme for searcher stereotype with tuple (t=0, p=100, e=20) in the second scenario

trend as before, where voting TF and TF schemes maintain a decent performance throughout the search episodes.

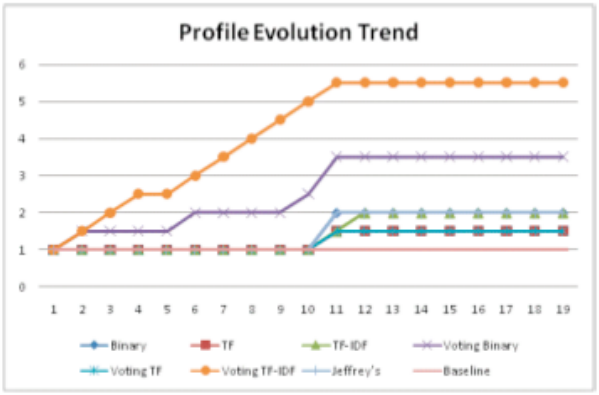

Figure 8.17: The profile evolution trend (PET) results of each term weighting scheme averaged for searcher stereotype with tuple (t=0, p=100, e=20) in the second scenario

Finally, I have averaged the recommendation precision across all search stereotypes in order to analyse the all-around performance of the system. Figure 8.18 presents the results in this aspect. We observe that TF weighting scheme starts with high recommendation precision, which however drops to very low levels after the drift occurs. The remaining models show an inferior performance during the first search iterations, but their performance also drops dramatically at marginal points. Voting TF-IDF and Jeffrey's conditioning weighting schemes have performed poorly. Throughout the simulation study, it was noticed that the performance of these models is unrelated to the underlying system configuration and experimental scenario.

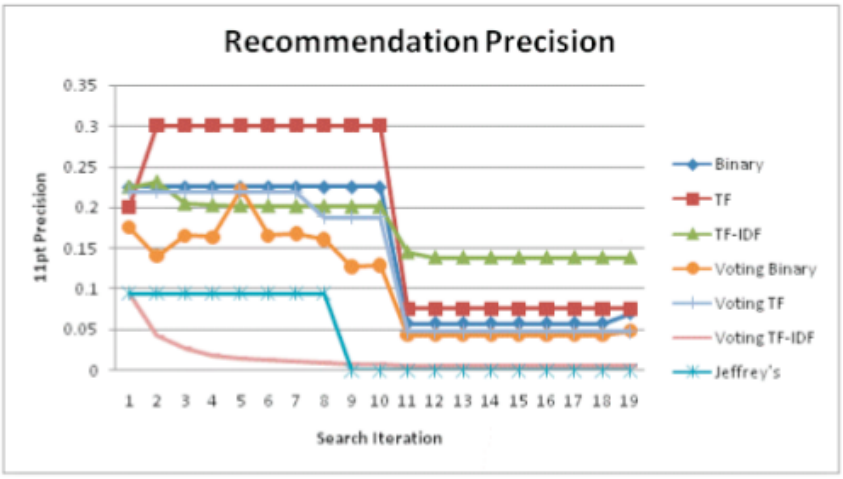

Figure 8.18: Average recommendation precision of each term weighting scheme averaged for all searcher stereotypes in the second scenario

Also, figure 8.19 displays the evolution of user profiles through the PET measure. TF and voting TF models are shown to outperform the remaining configurations, while maintaining a performance close to the baseline. As discussed earlier in this study, these weighting schemes have been observed to result in accurate profiles during the duration of this experimental scenario. In a similar fashion, Jeffrey's conditioning and binary weighting model perform very closely, yet far from optimal. On the other hand, TF-IDF and voting binary fail to detect the gradual concept drift, thus resulting in sub-par profile accuracy.

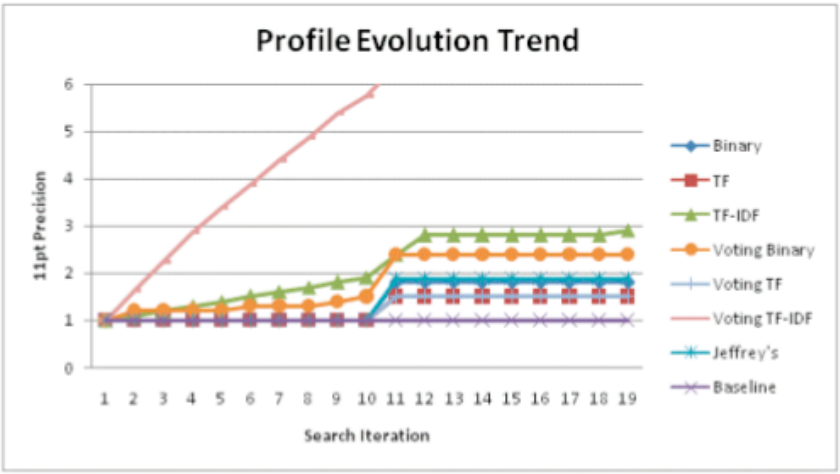

Figure 8.19: The profile evolution trend (PET) results of each term weighting scheme averaged for all searcher stereotypes in the second scenario

During the result analysis of this experiment, I came across a contradiction regarding the system performance in terms of average precision and profile accuracy. As far as average precision is concerned, TF-IDF has shown to outperform the remaining models for the majority of searcher stereotypes simulated. Although there existed weighting schemes that ranked higher for parts of each simulation run (e.g. figure 8.18), the influence in recommendation precision after the concept change has always been kept to low levels when compared to the remaining models. However, the results obtained through the PET measure has shown that TF-IDF fails to adaptively profile user needs. As shown in figure 8.19, the profiles computed by employing TF-IDF weights averaged to a maximum of 2.9 interests, while the optimal value was a single interest in the profile vector. A similar contradiction was observed for voting TF model, which topped the PET experiment, but remained unnoticed in the average recommendation precision test. Again, since no conclusions can be drawn given the current results, we will examine both weighting schemes in the next part of our analysis. The following section further evaluates the adaptive profiling process by examining the effect of a range of clustering thresholds.

8.2.2 Evaluating Clustering/Document Threshold Configurations

Since the underlying architecture of the profiling process is based on a K-Nearest neighbour algorithm, optimizing the performance of filtering thresholds is crucial. Cluster and document thresholds control the flow of new information and are responsible for maintaining an accurate snapshot of user interests. While we already examined the document recommendation and profiling process in the previous two scenarios, it is important to evaluate these components under a concept drift scenario. Based on information gathered during the previous part of this scenario, this experiment benchmarks a range of threshold pairs in terms of average recommendation precision and profile evolution trend.

In the earlier part of this section, I examined various the effect of various searcher stereotypes during the event of a simulated gradual concept drift. Models based purely on term frequency (TF and voting TF) stood out in the PET tests, but their results in the aspect of recommendation precision has not been exceptional. On the other hand, TF-IDF weighting scheme outperformed the remaining models in the recommendation precision tests and is ought to be further examined. Therefore, I have selected voting TF and TF-IDF models to examine in this subsection.

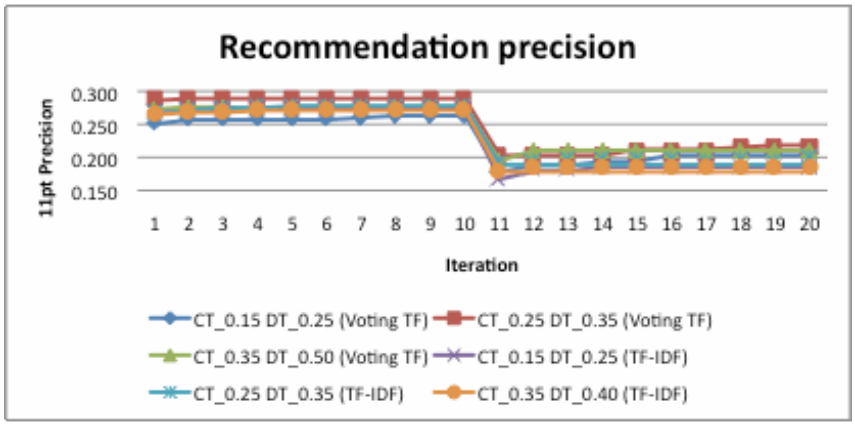

Figure 8.20: Average recommendation precision of a range of threshold settings averaged for all searcher stereotypes in the second scenario

As it is shown, both weighting models perform closely, whereas TF-IDF results have been identical

throughout the search iterations. Although the combination of voting TF scheme and a threshold pair of CT=0.25 and DT=0.35 appears to outperform the remaining configurations, it is shown that its performance has not been robust through the duration of this experiment. In general, the concept drift has largely affected the performance of all term weighting schemes, resulting in a sudden drop in recommendation precision, while very few schemes (e.g. voting TF) illustrates an increasing trend throughout search iterations.

8.3 Scenario 3

The third scenario involves simulation a concept shift, an abrupt change in the user's perspective or interests. Such rapid changes often occur when the user's interest rapidly shifts to a different topic, such as in breaking news. In essence, breaking news may attract the user's interests and abruptly change their search perspective for an unknown amount of time. As detailed previously, such temporal events can be very difficult to detect and categorize them effectively since they may become longer-term in the future, while more information are collected. The results of this experiment measure the performance of the profiling algorithm against the precision of document recommendations and the PET metric. Especially the latter is of exceptional importance in such profile-related experiments since it allows us to focus on the system's performance in marginal time intervals (i.e. the point of the abrupt change between topics). The remainder of this section is divided into two parts. Section 8.3.1 examines the effect of a series of weighting schemes, while section 8.3.2 studies the results obtained after attempting a range of threshold settings.

8.3.1 Evaluating Term Weighting Schemes

For the first part of this scenario, instead of focussing with a particular searcher stereotype, I have aggregated the results obtained through different searchers models into subsequent figures. At the same time, cluster and document thresholds are set to 0.25 and 0.35 respectively. Along this direction, figure 8.21 plots 11-pt precision values averaged and grouped by search iteration. Although several term weighting

schemes have performed quite closely to each other, there is a notional but distinct separation into three categories. The term frequency model stand out as the best performer in these results, while voting TF-IDF and Jeffrey's conditioning perform equally poorly throughout all search iterations. Similarly, the remaining weighting models resemble in the way they behave in this experiment, thus resulting in very similar performance.

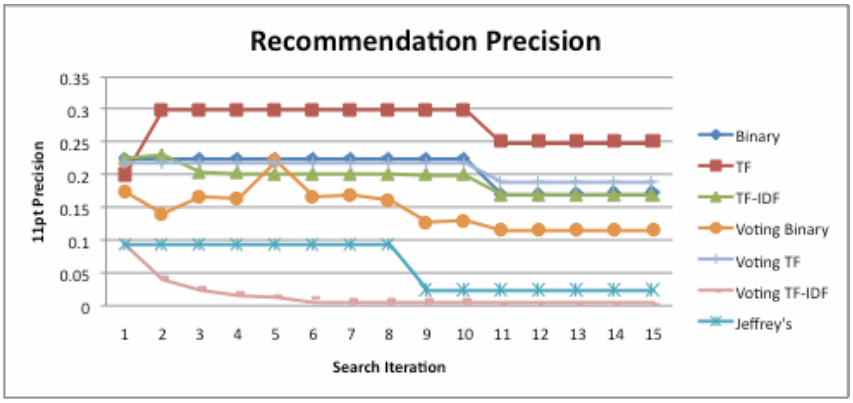

Figure 8.21: Average recommendation precision of each term weighting scheme aggregated for all searcher stereotypes in the third scenario

Also, I have employed the PET measure to examine the effect of the concept drift in the underlying profile structure along the whole duration of this experiment. The results along this direction are presented in figure 8.22. As in the previous case, the performance of term weighting schemes against the PET measure is very close to each other. Especially in the first set of search iterations the profile evolution trend for each model is almost indistinguishable. However, when comparing the profile of each scheme against the optimal baseline for this scenario, we observe models based on binary weights fall closer to the baseline. In addition, all term weighting schemes detect the abrupt change in concept, but with different rates of learning. For instance, it appears that voting TF has detected and accommodated the shift in perspective at the 10th iteration, which is the point that the concept shift occurs. On the other hand, the profile for each term weighting scheme coincides with the baseline at the start, but drifts away thereafter.

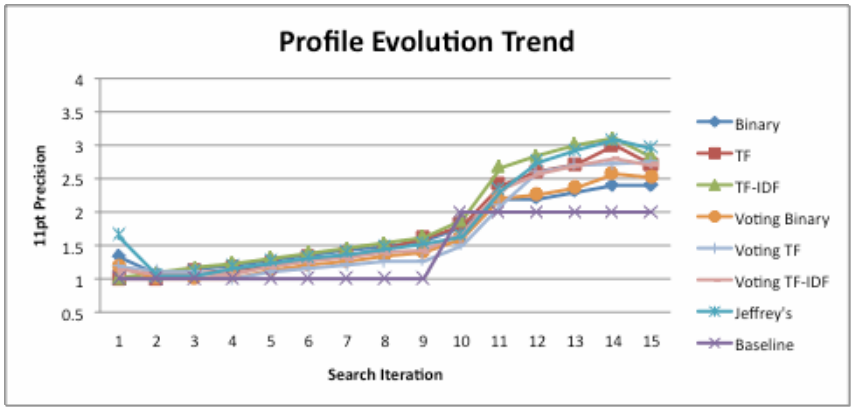

Figure 8.22: The profile evolution trend (PET) results of each term weighting scheme averaged for all searcher stereotypes in the third scenario

In order to analyse in more detail the outcomes of the PET measure, I have summarized the marginal differences at key search iterations and present them along table 8.4. The results for each search iteration are distributed into two columns; the left presents the PET measure at that iteration, while the right one shows the *marginal difference from the baseline* in italic. The best performer, the one closest to the baseline, for each search iteration is shown in bold. Based on these results, the voting TF outperforms the remaining weighting models at the marginal time points near the simulated concept change., while it also maintains a solid performance at the initial search iterations. On the other hand, the term frequency weighting scheme shows an exceptional performance in the first few iterations, but it rapidly deteriorates after the simulated concept shift.

Although a large series of runs were carried out for this experiment it was observed that all followed the same trend and was therefore decided to only show the aggregated results for all searcher stereotypes. Also, based on the results gathered and analyzed along this section, models based purely on term frequency (TF and voting TF) have illustrated a robust performance throughout the duration of this experiment. Term frequency has outperformed the remaining weighting models in the recommendation precision test, while the voting TF has produced an almost optimal profile under the PET measure. These schemes are further examined in the next section where I evaluate the profile capturing and document recommendation

	Search Iteration											
Weighting Scheme	1		2		8		10		12		15	
Binary	1.33	*0.33*	1.06	*0.06*	1.49	*0.49*	1.76	*0.76*	2.2	*0.2*	2.4	***0.4***
TF	1.00	***0***	1.00	***0***	1.45	*0.45*	1.81	*0.81*	2.62	*0.62*	2.72	*0.72*
TF-IDF	1.01	*0.01*	1.08	*0.08*	1.53	*0.53*	1.87	*0.87*	2.84	*0.84*	2.82	*0.82*
Jeffrey's	1.64	*0.64*	1.04	*0.04*	1.45	*0.45*	1.64	*0.64*	2.33	*0.33*	2.95	*0.95*
Voting Binary	1.18	*0.18*	1	***0***	1.34	*0.34*	1.61	*0.61*	2.21	*0.21*	2.52	*0.52*
Voting TF	1.17	*0.17*	1.10	*0.10*	1.25	***0.25***	1.52	***0.52***	2.17	***0.17***	2.75	*0.75*
Voting TF-IDF	1.12	*0.12*	1.06	*0.06*	1.37	*0.37*	1.66	*0.66*	2.68	*0.68*	2.69	*0.69*

Table 8.4: Marginal profile evolution trend (PET) difference between each term weighting scheme and the baseline averaged for all searcher stereotypes in the third scenario

component against a range of threshold configurations.

8.3.2 Evaluating Clustering/Document Threshold Configurations

In the earlier part of this scenario's analysis I focused on benchmarking the performance of Colombus in terms of recommendation precision and our bespoke PET measure. It was also shown that weighting schemes based purely on term frequency, TF and voting TF, result in significantly better performance than the remaining models. These two weighting algorithms will be further examined along this subsection.

Having benchmarked the adaptive profile capturing algorithm on the aspect of term weighting in a concept shift scenario, the next logical step was to assess the threshold configuration. Along this direction, the performance in terms of recommendation precision and profile evolution trend for a range of threshold settings were assessed and the results are presented below.

As in the previous part of this section I start by analysing the performance of the system in terms of recommendation precision. At this stage, I present average recommendation precision, for a range of threshold configurations, per search iteration aggregated for all searcher stereotypes. Figure 8.23 illustrates the results along this direction. Both term weighting schemes have resulted in very similar recommendation trend and performance. Similarly, although no significant (Wilcoxon test) difference

has been noted between threshold configurations, it appears that TF weighting scheme combined with a cluster/document threshold pair of 0.35/0.40 slightly stands out from the remaining configurations. The results of figure 8.23 has been analysed in more detail in table 8.5.

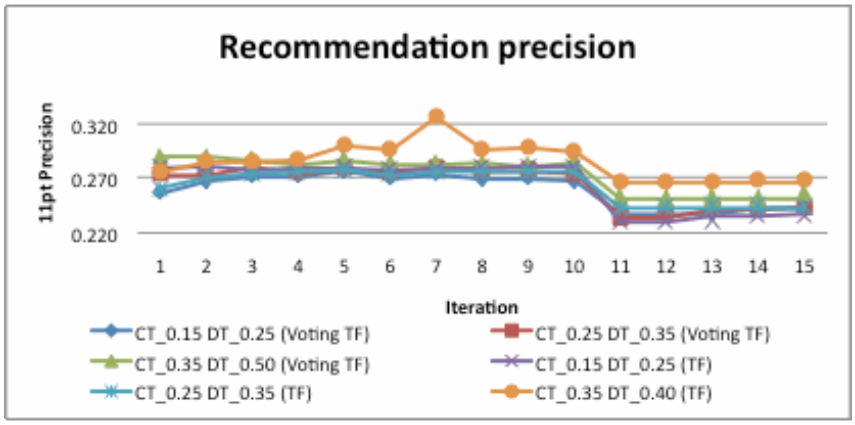

Figure 8.23: Average recommendation precision of a range of threshold settings averaged for all searcher stereotypes in the third scenario

Table 8.5 focuses on marginal points in the duration of search iterations to gain a deeper insight in the evolution of document recommendations with regards to threshold settings. The results for each search iteration are distributed into two columns; the left presents the recommendation precision at that iteration, while the right one shows the *marginal difference* from the previous point in italic. The best performer, the one with the best recommendation precision, for each search iteration is shown in bold. Supposing that this particular experiment is divided into two main parts, the one before the concept shift and the one after, we notice two trends in table 8.5. Voting TF has outperformed other configurations throughout search iterations that took place before the concept shift. On the other hand, non-voting TF has dominated the experimental results at the marginal point of the concept drift onwards. This subtle point illustrates the volatility of voting TF (and voting-based models in general), which is further explained at section 8.4. The common ground between the top performers constitutes the cluster threshold (0.35).

	Search Iteration									
Configuration	**1**		2		8		12		15	
CT/DT 0.15/0.25 (Voting TF)	0.257	-	0.267	*0.10*	0.269	*0.002*	0.242	*-0.027*	0.244	*0.002*
CT/DT 0.25/0.35 (Voting TF)	0.273	-	0.273	*0*	0.277	*0.004*	0.242	*-0.035*	0.242	*0*
CT/DT 0.35/0.50 (Voting TF)	**0.290**	-	**0.290**	*0*	0.284	-0.006	0.253	*-0.031*	0.254	*0.255*
CT/DT 0.15/0.25 (TF)	0.280	-	0.281	*0.001*	0.280	-0.001	0.230	*-0.250*	0.236	*0.251*
CT/DT 0.25/0.35 (TF)	0.261	-	0.271	*0.010*	0.276	*0.005*	0.245	*-0.029*	0.246	*0.246*
CT/DT 0.35/0.40 (TF)	0.276	-	0.285	*0.009*	**0.296**	*0.011*	**0.266**	-0.030	**0.268**	*0.002*

Table 8.5: Marginal recommendation precision difference between each term threshold combination averaged for all searcher stereotypes in the third scenario

8.4 Discussion

During the simulation study I came across a number of interesting findings. We mainly focus on the performance of Colombus under a number of different scenarios and employ the results previously discussed to derive an optimal configuration for our system. Along a similar direction, I also report a number of observations with respect to the correlation between the evaluation measures and the term weighting schemes or system configurations.

Aside from the performance-related observations, I also focused on finding any bottlenecks in the integrated algorithms. The profile learning scheme has received most of my attention due to its importance in the ongoing profile monitoring process. The main issues identified through this process are related to the profile learning and the threshold selection used throughout the clustering approach.

As detailed along section 6.5, the term extraction algorithm weights and ranks a set of interesting terms as identified through the relevance feedback process. Then, the profile learning scheme is responsible to update the user's profile with this information. Future search episodes are categorized as relevant to the existing interests based on the context of new information; if the extracted terms are similar to an existing interest, then these objects are merged and the term weights are updated. So the quality of the user profiles is bound to be affected by the extracted term set from the current search iteration as well as the threshold setting that essentially designates the flow of new information.

When comparing a set of new information against an existing interest, the similarity matching function will check that their underlying distance is not below a certain threshold. It has been observed that, although the extracted terms from a search iteration can be conceptually on the same topic as an existing interest, information can flow and create a new facet in the user profile. For instance, suppose a user profile with one interest with term vector as follows: { buy[4], car[4], new[3], used[2], dealer[2], fast[2] } where the underlying weights are shown in in square brackets. Also, suppose the query "find bmw convertible" returns the following term set: { car[3], find[3], bmw[2], convertible[2], model[2], sale[1] }. The cosine distance between the two term sets is *0.296* which indicates that the term set will be inserted as a new interest in an existing category in the user profile (assuming a threshold pair of 0.25, 0.35). However, it is obvious that the search the user has initiated was on the same subject as the interest in his profile. In essence, a term set that contains synonyms of the terms contained in an existing interest may be treated as a new search need. The rest of this thesis refers to this observation as the "synonym effect". This issue has been observed especially in newly created interests that contain only few information in the underlying term vector.

In order to overcome the synonym effect the profile learning algorithm must continuously remodel the user's profile and readjust the number of interests by separating distant object and, more importantly, merging similar facets. Although Colombus handles the latter by constantly monitoring user profiles, there is no guarantee on the amount of time that it will take two interests about the same subject to converge as far as their term vectors are concerned. Also, this process will be quite time-consuming on large-scale profile-aware systems with thousands of users and interests. The profile learning approach needs to calculate the distance of every interest with each other and also take into account any merging

and splitting operations. A few ideas of possible ways around this issue are presented in chapter 12 - Future work.

Another interesting observation was discovered when comparing the correlation-based measure against the recommendation performance under that particular configuration. I noticed that there is only limited correlation between these two aspects; a term weighting scheme with exceptional correlation performance doesn't necessarily rank exceptionally in terms of recommendation precision. Jeffrey's conditioning model confirmed this observation as it is shown in figures 8.1, 8.4, 8.9, 8.13, 8.18 and 8.21. Also, the TF-IDF scheme has outperformed the remaining models in several recommendations precision tests (figures 8.7,8.13 and 8.16), but performed quite poorly in profile learning related tests such as the PET (figure 8.19). However, since the focus of Colombus is document recommendations *through adaptive user profiling*, profile accuracy has a very high weight in the results analysis stage compared to the alternative measures.

As far as the comparison of term weighting schemes are concerned, we observed difference between voting and non-voting weighting schemes in the way they behave in noisy environments and learn from new information. A noisy environment is defined as relevance feedback set containing both relevant and non-relevant documents. This can be produced when a user visits both relevant and non-relevant documents during a search episode, thus implicitly judging non-relevant documents as relevant. It was observed that voting schemes were much more volatile in the sense that they can absorb and exploit new information more effectively than non-voting alternatives. Also, in concept shift/drift scenarios, along the last two experiments, voting schemes showed a faster rate of learning near the marginal point of concept change. This effect was illustrated in sections 8.2.1, 8.3.1 and figures 8.4 and 8.9in particular, where we observe that voting-based term weighting schemes have been among the top performers.

However, such volatility also translates to less resistance to noise. As it was illustrated in figures 8.1 and 8.4, the performance of such schemes, especially voting TF, has shown an exceptional performance compared to the other weighting models. The same scheme's performance has slightly deteriorated in experiments carried out in much more noisier environments like the ones shown in figures 8.8 and 8.16. Especially in the latter figures, non-voting schemes have outperformed their voting counterparts, illustrating their higher resistance to noisy environments. Although voting TF is still shown to outperform

several non-voting alternatives, its average recommendation precision has dropped much more along the course of search iterations.

With regards to the threshold settings, we observed that there is no direct correlation that vastly changes the results obtained. When comparing correlation-based measures against the same ones obtained under different threshold settings we observe the same trend in the result set. Figures 8.10, 8.20 and 8.23 confirm these findings.

Based on the results analysed and presented throughout this chapter it appears that an adaptive retrieval model based on a voting TF weighting scheme and a threshold combination of 0.25/0.35 for cluster and document threshold respectively will optimise the system's performance. In fact, this particular has been employed during our user evaluation study to calibrate both Colombus and SearchIT, our baseline system. A thorough analysis of the experimental procedure for the user study is included as part of chapter 9, while the evaluation results are presented in chapter 10.

Chapter 9

User Experiments

9.1 Introduction

The simulation-based study presented in the previous chapter was carried out to evaluate Colombo in terms of the retrieval effectiveness of document recommendations and adaptivity of user profiles. After analyzing the results obtained through this simulation study I discovered that the voting term frequency weighting scheme combined with cluster threshold of 0.25 and a document threshold of 0.35 outperformed the remaining system configurations. The weighting models and threshold pairs were evaluated over a variety of experimental scenarios for a series of searcher stereotypes. The aim was to obtain a realistic indication of the system's performance by simulating user search behaviour under several common scenarios and environments. However, even such large-scale simulation studies exclude the user from the evaluation stage and fail to study subjective aspects. Therefore, there is an apparent need for carrying out real, interactive experiments in order to investigate the behavior of our system under the constrains imposed by real users.

Only few research studies evaluating relevance feedback systems have studied users and were conducted in operational environments (Hersh & Hickam 1995, Efthimiadis 2000, White, Jose & Ruthven 2003). Driven by the lack of established evaluation methodologies for adaptive IR systems, I have designed and

carried out a task-based user-centered experiment adopting several concepts presented in (Borlund & Ingwersen 1997, Psarras & Jose 2006). The chapter begins by describing the experimental procedure followed throughout this study, while the description and role of each experimental task are presented in subsequent sections.

9.2 Experimental Subjects

During the recruitment process, a total of 15 participants agreed to participate in this experiment. Recruitment was carried out by sending an "invitation to participate" to the electronic mailbox of students in the University of Glasgow. The experiment was advertised through large capacity mailing lists in the department of computing science. Non-student invitations to friends and family invitations were also sent. The final sample of participants was chosen from a pool of volunteers on the basis of their search experience and demographics, but an attempt was made to maintain a balance between the number of students and non-students.

The recruitment process targeted participants from diverse education, backgrounds and computer skills. Participants were categorized to experienced or inexperienced on the basis of their answers in the entry questionnaire. The experienced searchers were those who used computers and searched the web on a regular, often daily basis. Inexperienced searchers were those who both searched the web and used computers and the Internet infrequently. Past research (Hoscher & Strube 2000) has shown that experienced computer users conduct their web searches differently. Therefore, one of our objectives was to examine whether evaluation trends, such as usage patterns, can be observed for both experienced and inexperienced searchers.

The average age of experimental subjects was *24.7* years, ranging from a minimum of *22* to a maximum of *31* years. Also, *87%* of participants had or were pursuing a university degree, while *35%* had studied or were studying computer science. Also, the subjects were divided into two groups, experienced and inexperienced, on the basis of their answers in the entry questionnaire. Table 9.1 reports the classification criteria from a number of likert scales used in this sense. The percentage of each user category (experienced/inexperienced) is presented within parentheses.

Factor	Inexperienced	Experienced
Number of experimental subjects	7 (3 male, 4 female)	8 (4 male, 4 female)
How often do you carry out online searching at home or work?	'1-2 times a week' (73%)	'Several times a day' (88%)
How often do you find what you are looking for when using search engines?	Sometimes (83%)	'Very Often' (92%)
How often you search repeatedly for the same or similar topics?	Often (54%)	Sometimes (55.5%)

Table 9.1: Inexperienced and experienced subject characteristics

9.3 Experimental Tasks

Our experimental tasks were designed to evaluate the effectiveness of two information seeking portals in a concept changing environment. Also, measuring the degree of adaptivity, as well as the accuracy of user profiles, was another major target of this experiment. There are two types of tasks that I considered for our evaluation study: open-ended, where subjects are free to search about their personal interests, and common, shared tasks, which are structured by the evaluator and are common for all subjects. After conducting a user study (Psarras & Jose 2006) on a similar recommender system, it was discovered that open-ended tasks are less suitable for our objectives. On the other hand, a common task allows us to check and compare the adaptive behavior of the system in a controlled environment, while personal interests of users (open search tasks) cannot be equally evaluated.

My objective was to create an environment where subjects would interact with experimental systems, as if they are performing their own searchers. To facilitate this effect, topics were placed within simulated situations as proposed in Borlund (Borlund 2000, Borlund & Ingwersen 1997). The technique asserts that searchers should be given search scenarios that reflect and promote a real information seeking situation. An example along this direction is presented in figure 9.1. Also, I expect to see people exploring different facets of their tasks, since the topics have been chosen on purpose to be quite generic.

Task Context:
You recently graduated from university and are looking to start a job in the industry. As you don't have any previous experience in this aspect, you are looking to collect more information with regards to the interview process.
Task Description:
Use two information retrieval tools to find more information with regards to this topic.

Figure 9.1: An example task briefing

In our proposed methodology, part of the experimental tasks were common for all users. In particular, two common tasks for all participants were associated to each experimental subjects, drawn from recent breakthroughs in technology. The proposed common tasks are presented in table 9.2:

Furthermore, another pair of tasks was included in our experimental design to allow users to explore their personal preferences and criticize the system's performance more actively. Each experimental user was asked to choose two or more of his own interests and hobbies to find more information through Colombus and SearchIT systems. Past evaluation studies (Psarras & Jose 2006) have illustrated that experimental subjects are more attracted to search for their personal tasks, while they tend to ignore the rest.

Finally, White (White et al. 2005) suggested that the order of appearance for experimental asks may constitute an influencing factor in evaluation studies. Therefore, an appropriate policy was designed to alternate the order of tasks for experimental subjects.

9.4 Experimental Systems

Two experimental systems were used for the purposes of this evaluation. Both systems share the same document indexing and retrieval component, as well as search interface, but greatly differ in terms of profiling and recommendation policy.

Task Title	Task Description
Job search	You recently graduated from university and are looking to start a job in the industry. As you don't have any previous experience in this aspect, you are looking to collect more information with regards to the interview process. Use two information retrieval tools to find more information with regards to this topic.
iPhone	You are a student in Glasgow university and you have been given an essay to write about the recent release of iPhone. The essay title is "iPhone: The future of mobile devices". Use the two retrieval tools to find more information with regards to this topic.
Traveling in Europe	After successfully completing a stressful project at work you have given a week off. You are dreaming a luxurious trip around Europe, but unfortunately you only have limited money to spend. Use the two retrieval tools to find a cheap destination for the 1st week of the following month.
Drugs and steroids scandal	After winning numerous medals in the latest Olympic games Marion Jones have been forced to forfeit all her medals, points and prizes won during her career. You are a local journalist asked to write an article about the use of steroids by athletes. Use the two retrieval tools to find more information.
Buying a property	You are a local businessman in the Glasgow, UK (choose any city/country you wish) thinking of buying a property in the next few months. However, after the latest interest rates drops and the market slowdown you are thinking that the long-awaited house market price drops are only a few months away. Use the two retrieval tools to gather more information on this matter and avoid rash decisions.

Table 9.2: The list of proposed topics for the user study

9.4.1 Colombus

A full-engineered version of Colombus was used as the main experimental system. As far as the underlying retrieval engine is concerned, Colombus communicated with Google through Google API to retrieve relevant documents given the user's query. Since TREC engine does not offer indexing and searching capabilities for documents in the world wide web, it was considered unsuitable for this part of our evaluation.

9.4.2 Baseline (SearchIT)

Throughout the user experiments, I deployed a second system in order to examine the necessity of an adaptive information assistant in the search habits users. This baseline system interfaces with Google news alert, a basic recommendation system available in the web. The baseline system was designed around Google news alert API and labelled SearchIT to avoid any result bias against any of the search systems. Another reason for designing a wrapper around Google news alert was to monitor user actions and analyze them to discover the trends throughout the entire experiment.

Colombus and SearchIT used the same interface components and retrieval algorithm but differed in the fact that SearchIT only contains the functionality of a web search engine. No personalization, customization or profiling features are available, so the each experimental subject was solely responsible to generate an appropriate set of query terms and find relevant documents with respect to the experimental scenarios and their needs. Figure 9.2 illustrates a sample search result page in the baseline system.

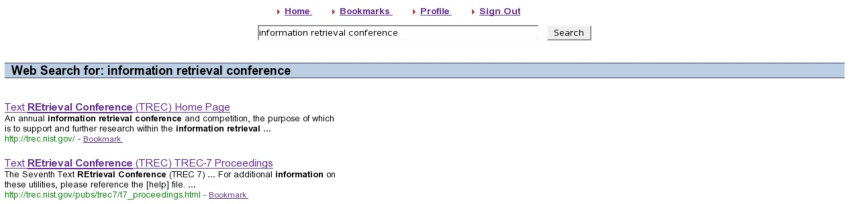

Figure 9.2: The SearchIT search interface, illustrating a sample results page

9.5 Experimental Hypothesis

Our main hypothesis is that Colombus can capture evolving user information needs and pro-actively fetch relevant information, thus becomes a personal information assistant that can satisfy user requirements effectively. The main hypothesis can be split in smaller components in order to directly relate to the questionnaire and log data recorded:

1. Columbus is an effective search system

(a) It is possible to identify the facets of user need

(b) Adaptation improves search effectiveness

2. We can capture user evolving needs

   (a) Combination of implicit and explicit feedback can model evolving user needs

   (b) Users avoid editing their profiles very often

3. Personal Information Assistant can satisfy user requirements effectively

   (a) It can fetch additional relevant documents

   (b) Presenting information on a priority basis is appropriate and effective

9.6 Experimental Procedure

The user study was carried out in the following manner: Initially, participants were informed on the features and capabilities of our system. Upon agreement of participation, they were asked to complete an entry questionnaire with respect to their background as well as their search experience on the web. Their answers were mainly used to classify them to experienced/inexperienced and categorize them in relevant age groups. Then, they were given the description for all general search topics.

This evaluation experiment lasted a total of 14 days, such that relatively long-term search intentions of users were captured. Participants were invited to use both systems during this period of time to complete the experimental tasks. This way I was able to monitor the usage pattern for each system in order to record user preferences over time. Also, on 5 day intervals, each experimental user was asked to complete a *post-search* questionnaire to capture their views on each system. At the same time, changes to the common task description were communicated to each participants. Communication throughout the experimental study was conducted mainly through emails.

9.7 Data Collection

9.7.1 Questionnaires

Questionnaire data was the main method used to record user thoughts on particular aspects of the system. A combination of likert scales, semantic differentials and open-ended questions were employed. A total of three questionnaires were designed and completed by the users at different time intervals. These are included in appendix B. Apart from *entry* and *exit* questionnaires, another one was designed to capture the user opinion during the experiment. Along this direction, users were asked to complete this *search* questionnaire twice, after 7 and 14 evaluation days. Although the aim of each questionnaire was different, there was a common focus towards capturing the effectiveness of document recommendations and profile accuracy.

9.7.2 System Logging

Quantitative data was also collected through extensive background logging, used throughout the duration of the evaluation period, to record user interactions with the system. Apart from retrieving general statistics about the profiling procedure, our aim was also to answer several research questions, mainly focusing on concept drifts and adaptive filtering. The questions I attempted to answer through background logging are divided into three main categories:

9.7.2.1 Profile-related questions

- How the profile evolve?
- How many times a profile was changed?
- How many words are added/deleted explicitly?
- How many times a recommendation was accessed/deleted/bookmarked?
- How many times a recommendation was accessed/deleted/bookmarked per day?

9.7.2.2 Task-related questions

- How many tasks did the users attempt? (Can be also retrieved explicitly through the exit questionnaire)
- How many interests were created in their profile from these tasks?

9.7.2.3 General questions

- How many searches were issued
- How did user interact with search results (click-through, view summary etc.)
- Is there any change on their search pattern (number of terms used, number of search results accessed)
- Are these changes related to the evaluation period? (1-4 days, 5-9 days, 10-14 days, 15-20 days)
- What system do they use more?
- How does the system access pattern evolve over the duration of the evaluation period?

Chapter 10

Results and Analysis

In this chapter I present the results of the user study against Colombus and SearchIT, the systems used throughout the experiment. Experimental results are classified in three main categories: Profile capturing, recommendation performance and system preference. While on the first two parts I directly focus on the performance of Colombus, on the third section I present the user preference when asked to choose between Colombus and SearchIT. I also focused on proving or disproving the evaluation hypothesis illustrated in section 9.5 and present the result along section 10.4. The results are analyzed in terms of questionnaire data and background logging and summarized in tables and figures.

10.1 Profile Capturing

The degree of adaptivity of a profile aware information portal is directly related to the amount of time it takes to detect concept drifts and adjust to them accordingly. During the simulation experiments I employed a custom metric to measure the performance of the profile learning algorithm and analyzed the results to calibrate Colombus. The results from this user study will allow us to gain more insight in this particular aspect of the system by taking into account user experiences after having used Colombus.

Figure 10.1 aggregates the data collected after 7 and 14 days respectively with regards to the accuracy of

user profiles. Questionnaire data collected after 7 days of using the system indicate that even at the start of the learning process the system is able to profile user needs effectively. At the same time, the profile accuracy according to the unique demands and standards of experimental subjects shows an increasing trend as the learning process gathers more information. In particular, after 14 days of using the system, 75% of the users are satisfied with the profiling performance of our system.

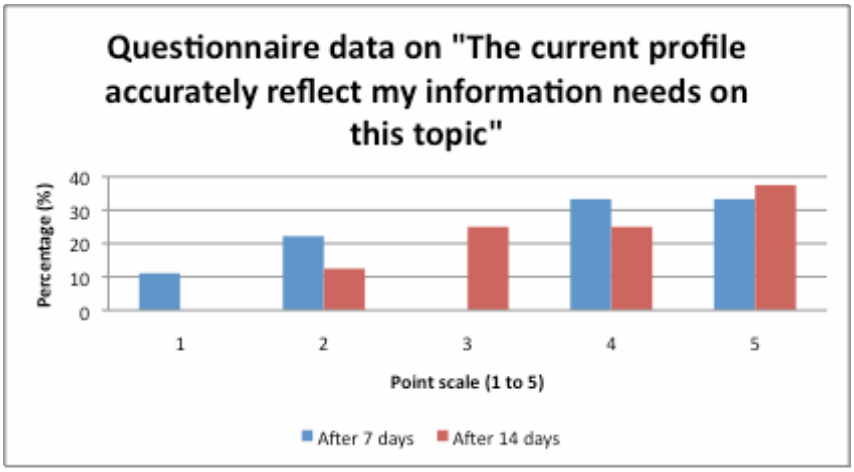

Figure 10.1: Questionnaire data on how accurate user needs are reflected on their profile

As detailed in earlier parts of this thesis, profiles cannot be simply decomposed to individual interests. Due to the nature of user needs, profile interests are multi-faceted, such that they contain several sub-topics. I have attempted to capture this subtle point by measuring the effectiveness of the profile learning scheme as a whole and not of individual topics as presented in figure 10.1 earlier. Questionnaire data captured again at two evaluation intervals (after 7 and 14 days) illustrate that, towards the end of the user study, the vast majority (88.8%) of experimental subjects appeared to be satisfied with the way their interests were recorded. Also, there is a significant difference between user responses given after 7 and 14 days as illustrated in figure 10.2, which indicates that user profiles become more accurate over time.

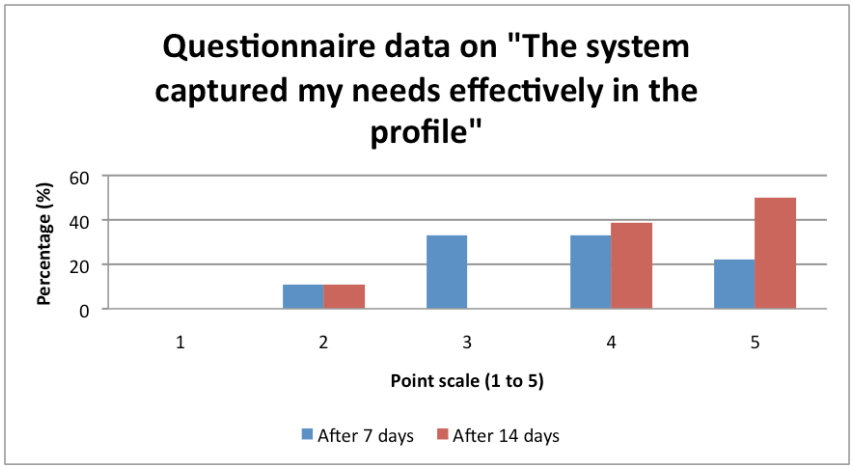

Figure 10.2: Questionnaire data on how effectively user needs are captured in profiles

Furthermore, the performance of the profile learning scheme is reflected through explicit changes users have to make to accurately represent their interests and needs. A change in a profile is translated as an explicit addition or deletion of an interest or alternatively a change in the interest contents. We can safely assume that a profile edited continuously throughout the user experiments indicates an unreliable profile learning stage. On this aspect, data collected through background logging are presented in figure 10.3. Data are categorized in three aspects: Interests added or deleted or amended through the profile management interface. As illustrated in the chart below, the average of profile changes is kept at very low levels (less than 0.50 changes) throughout the duration of the user study. Also, there is a decreasing trend of profile changes which converges to zero closer to the end of the evaluation period. Ideally, the aim is a decreasing trend in explicit profile changes and an increasing trend in document recommendation access. This observation will be revisited further below when I discuss the results of this user study.

Data along this direction was also retrieved through questionnaires. Experimental subjects were asked to indicate the number of times they had to manipulate their profile explicitly. Approximately 80% of the participants thought that it wasn't required to intervene in the profile learning stage, however, no significant change was observed in data collected between 7 and 14 evaluation days. The average likert

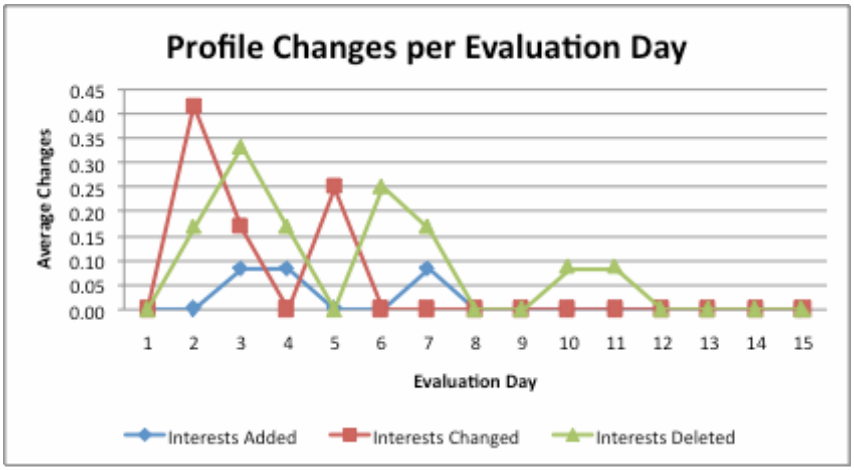

Figure 10.3: Explicit profile changes per evaluation day averaged for each experimental subject

scale rating retrieved after 7 days was 2.11, while after 14 days it dropped to 1.99 (lower is better). Figure 10.4 illustrates the results along this direction. <invert chart>

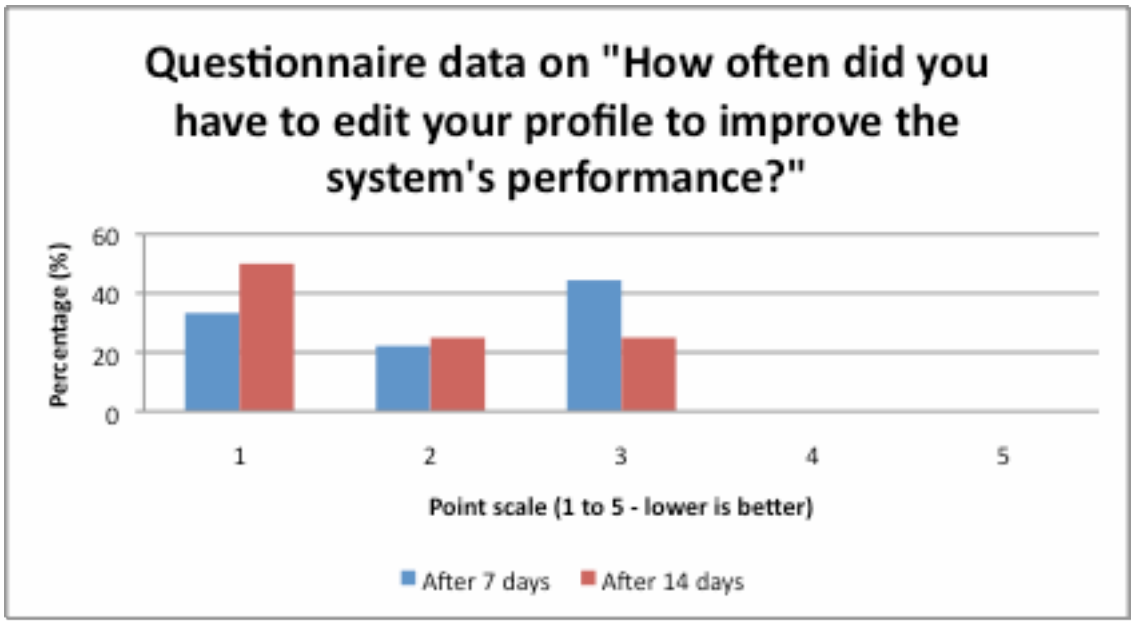

Figure 10.4: Questionnaire data on explicit profile changes during the user study

Last but not least, experimental subjects were asked to rate the adaptivity of the system in tracking and accommodating changes with regards to their search preferences. Although, searchers may lack the required experience to detect conceptual drifts in their personal interests, the results along this direction allow researchers to get some feedback with respect to the underlying learning scheme.

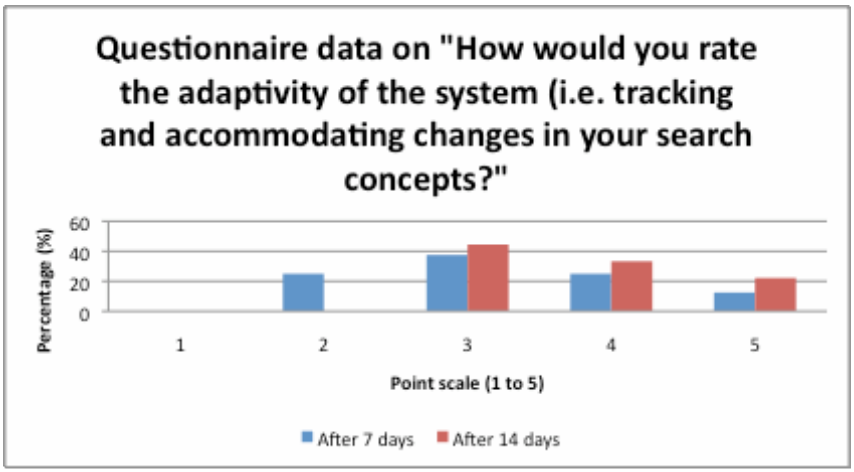

Figure 10.5: Questionnaire data on the adaptivity of Colombus during the user study

As figure 10.5 indicates, there is a mixture of opinions after the first evaluation days resulting in an average rate of 3.25. Similarly, after 14 evaluation days, 55% of participants judged the degree of adaptation satisfactory resulting in a rating increase to 3.78. On the other hand, although the remaining participants found there is still room of improvement in this area, the learning algorithm can only make qualified decisions based on the amount of information it has at that time. The observed trend shows that as experimental subjects use the system for their everyday searching tasks, the effectiveness of the profile learning scheme increases analogously.

10.2 Recommendation Performance

As the ultimate goal for a recommendation system is to discover documents that satisfy the users' information needs, I will first cover the evaluation results of this particular aspect of the system. The results are analyzed in terms of questionnaire data and background logging and summarized in the following tables and figures. Along this direction, I asked the users to judge the performance of the system in finding document recommendation on their behalf. The results are presented in figure 10.6.

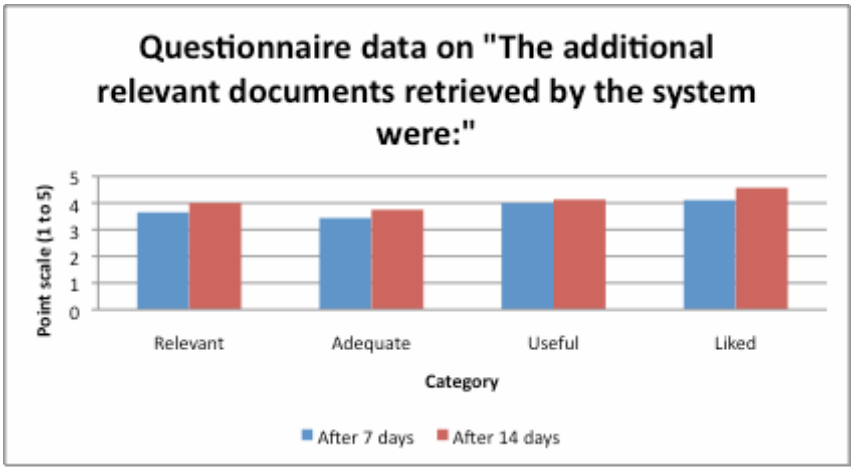

Figure 10.6: Questionnaire data on the effectiveness of document recommendations during the user study

According to the experimental subjects, the performance of the system in retrieving relevant documents has been kept at high standards throughout the whole evaluation period. More importantly, there is an increasing trend in all categories between the capturing intervals. Also, although the difference between each interval seems quite unimportant, there is always an increase in the underlying numbers when comparing data captured after 7 and 14 days.

Furthermore, the aspect of recommendations attractiveness has also been benchmarked during Colombus user study and the results are illustrated in figure 10.7. Near the end of the evaluation study, almost 80% of the users have found interesting documents through Colombus, while all participants have declared

that they found at least one piece of unseen information for each of their interests through our system (figure 10.8).

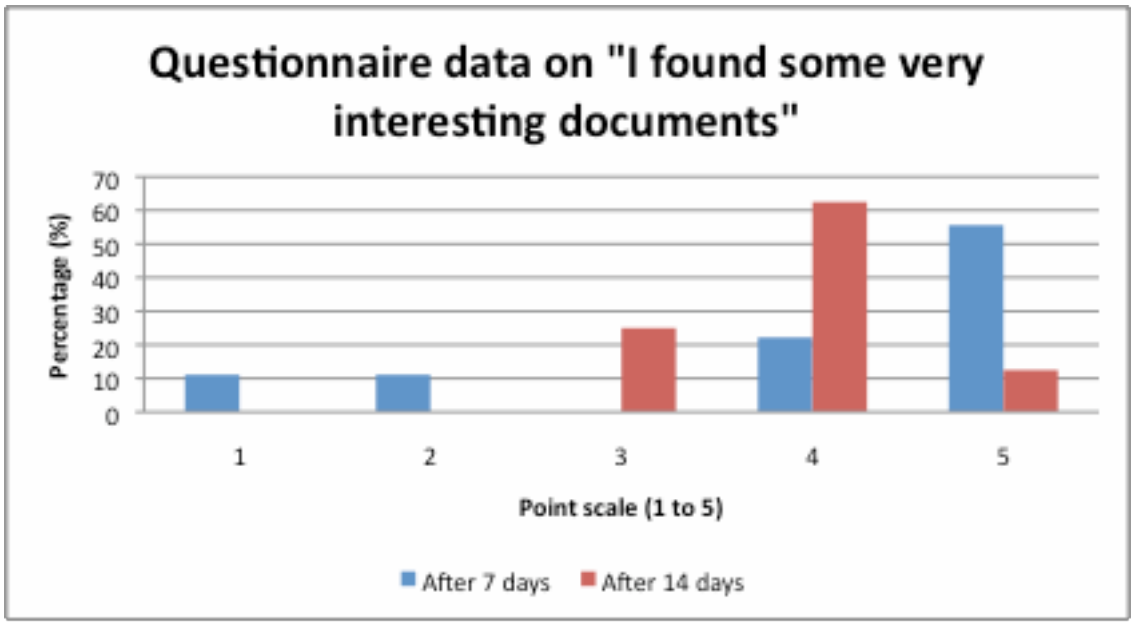

Figure 10.7: Questionnaire data on the interestingness of document recommendations

The effectiveness of document recommendations were also assessed by analysing data collected through background logging. Figure 10.9 presents the recommendation access per evaluation day averaged for each of the experimental subjects.

10.3 System Preference

As detailed along section 9.4.2, a second system, identical to a modern search engine both in terms of look and feel but also retrieval performance, was used as a the baseline for our hypothesis. The results

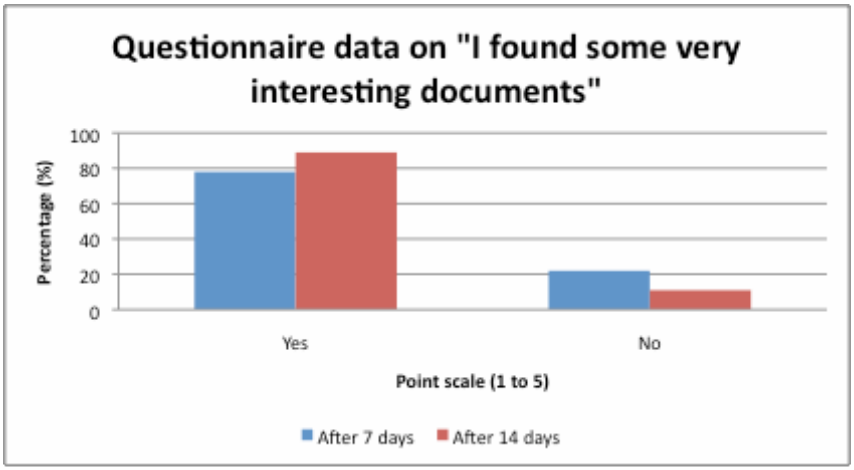

Figure 10.8: Questionnaire data on the novelty of document recommendations during the user study

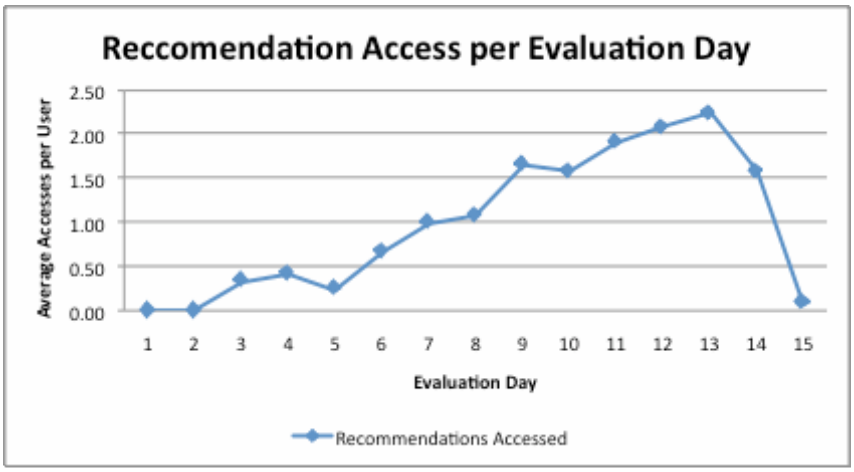

Figure 10.9: Recommendation access per evaluation day averaged for each experimental subject

of questionnaire and log files have been analyzed and are presented along this section.

Both Colombus and SearchIT were promoted and explained in a very similar way and since there is no significant difference in the user interfaces, both systems should exhibit a similar usage pattern. However, these systems are greatly different in the services they provide. Colombus is an adaptive document recommendation tool, while SearchIT is only a search engine. As detailed in section 9.4.2, SearchIT employs the same search component and interface integrated in Colombus, so the search results of a

given query are identical for both tools. Figure 10.10 illustrates the average number of searches issued in each system grouped by evaluation day, which indicates the implicit user preference with regards to the systems used in this study.

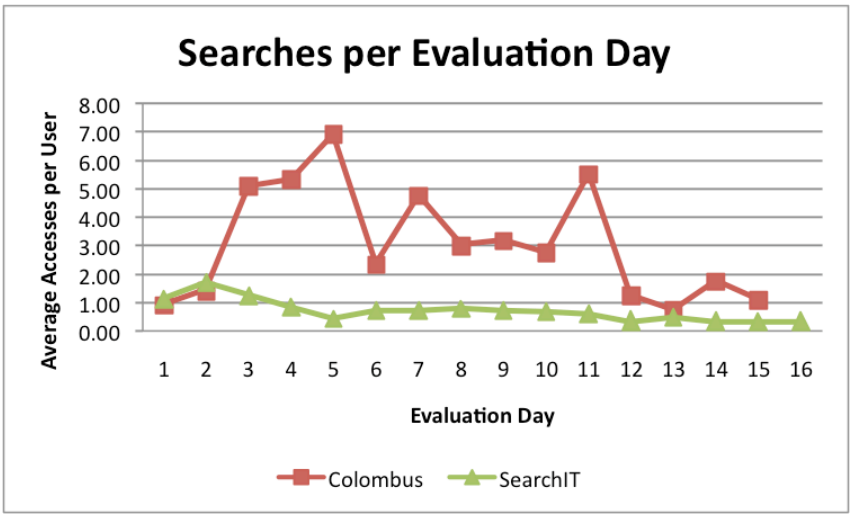

Figure 10.10: Searches issued per evaluation day averaged for each experimental subject

Although experimental subjects appear to have issued a similar amount of searches to both systems initially, Colombus access pattern has increased significantly after the third evaluation day. In effect, experimental subjects trusted Colombus more for their daily search needs, especially after their profile has been fully created and the document recommendations start to appear. On the other hand, the access pattern of SearchIT has shown a significant decrease from day 2 to 5, while Colombus rapidly increases at the same time. During the remainder of the evaluation period, experimental subjects issued less than a search on average through SearchIT.

Regarding the performance of each system, part of the exit questionnaire data captured user views on how well each system performed. Experimental subjects were asked to rank Colombus and SearchIT on a scale 1 to 5 (5 being the highest) taking into account the performance of each system throughout the evaluation period. Over 80% rated Colombus above average, while at the same time, SearchIT was graded as ineffective by almost 50% of users. The results are presented in figure 10.11.

Similarly, the same questionnaire asked users how much each system helped them in their daily search

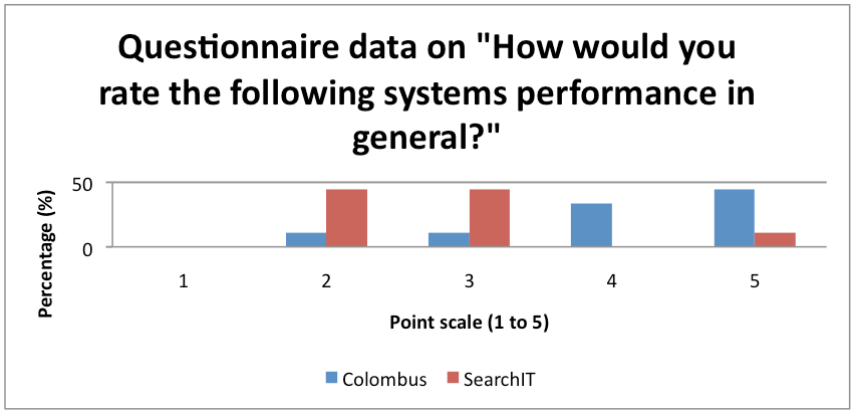

Figure 10.11: Questionnaire data on "How would you rate the systems performance in general"

experience throughout the duration of the user study. Data retrieved along this direction is presented in figure 10.12. The results appear to be more diversified than those presented in figure 10.11 earlier, but Colombus still maintains a solid performance. More specifically, one out of two users found a benefit in using Colombus for their daily search needs, while 80% of experimental subjects ranked the system with a non-negative grade in the point scale. Ont the other hand, SearchIT results are significantly worse than Colombus, with over 40% of users voting that their daily search experience has deteriorated.

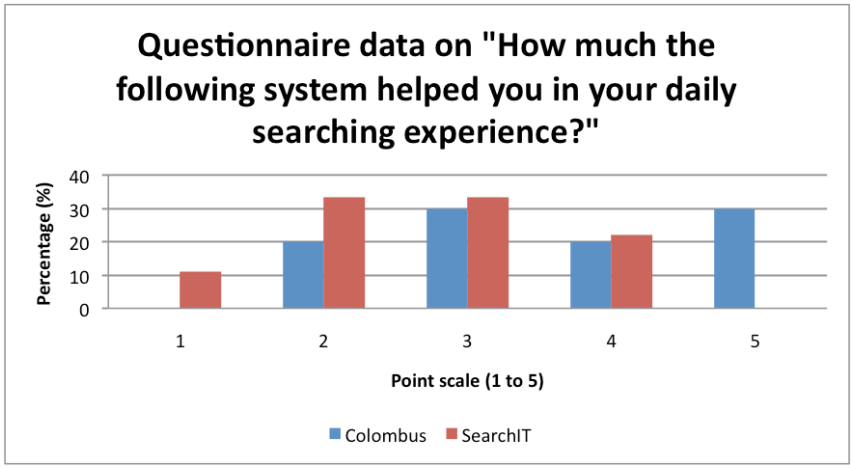

Figure 10.12: Questionnaire data on "How much the following system helped you in your daily searching experience"

Also, since commercial search engines currently dominate the web, it is important to capture the views of

experimental subjects when measuring the effectiveness of their favorite search engine against Colombus and SearchIT. Data collected along this direction have been analysed and are presented in figure 10.13. The results show that Colombus is favoured by the majority of experimental subjects over SearchIT and their other preferred search tools. More than 50% of the experimental subjects found Colombus better than their favorite search engine, while only 10% approximately have disliked it. The results follow a dissimilar trend for SearchIT, where only 11% of experimental subjects found the system better compared to the tool they use for their daily searching needs.

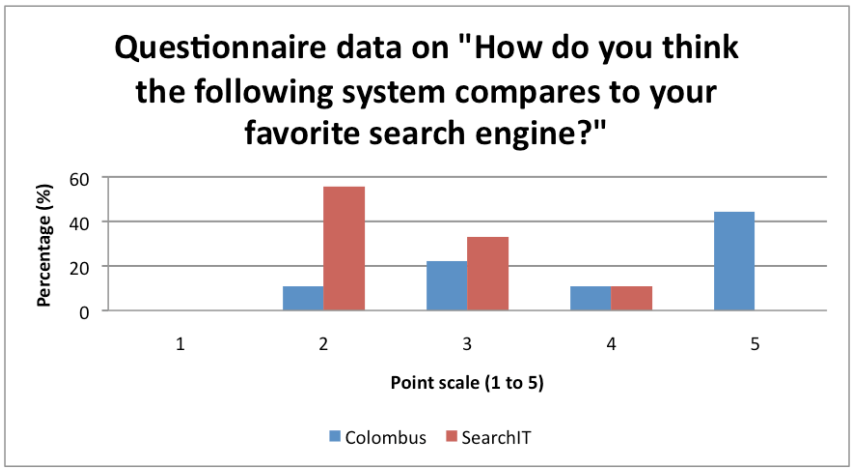

Figure 10.13: Questionnaire data on "How do you think the following system compares to your favorite search engine"

Furthermore, another part of the exit questionnaire captured the responses of experimental subject by directly comparing the two systems in an absolute scale (disagree vs. agree). Figure 10.14 illustrates the results of whether users were able to satisfy their search requirements with each of the two systems. The results prove our initial hypothesis, that an adaptive profile-aware information filtering tool can certainly improve the user search experience, an observation confirmed by approximately 90% of participants.

The direct comparison of Colombus versus the baseline system was completed by capturing the user preference with regards to which tool they liked more. The results of this question, illustrated in figure 10.15, are very similar to those presented in figure 10.14 earlier. The majority of experimental subjects (88.8%) have indicated their preference in Colombus over SearchIT, which further enhances our original

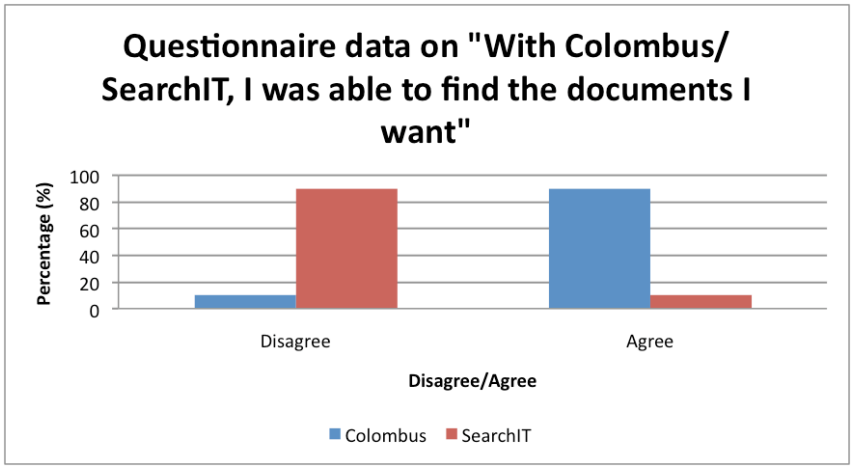

Figure 10.14: Questionnaire data on "With Colombus/SearchIT, I was able to find the documents I want" hypothesis.

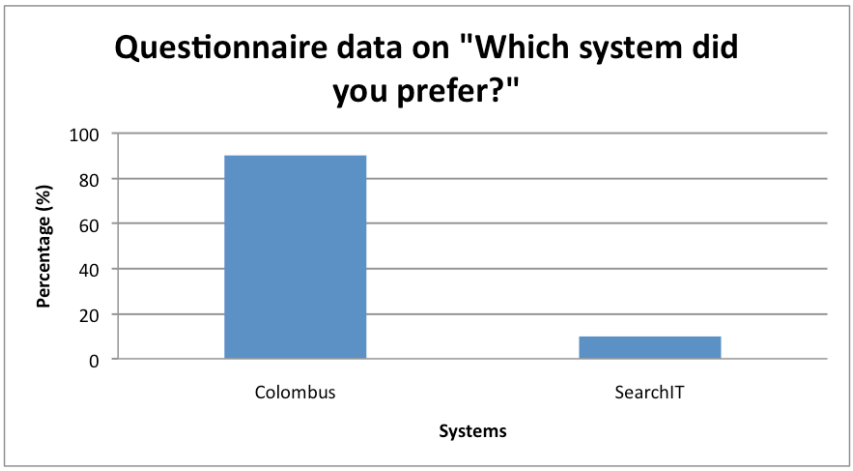

Figure 10.15: Questionnaire data on "Which system did you prefer"

10.4 Evaluation Hypothesis Analysis

In this section I discuss the outcomes of the evaluation and relate them to our original hypothesis presented along Section 9.5.

1.a It is possible to identify the facets of user needs

On figure 10.2, I presented the results of questionnaire data regarding the accuracy of user profiles. Experimental subjects indicated that they were very satisfied by the contents and overall state of their profile. More specifically, users rated the profile learning scheme with 3.66/5 and 4.27/5 points, after 7 and 14 evaluation days respectively. At the same time, near the end of the user study, more than half of the participants graded the profile learning scheme with the maximum points, while approximately 90% of experimental subjects assigned a positive score to this question.

Other results to support this part of the hypothesis have been illustrated along section 10.1 and shown that user profiles in Colombus are captured accurately and effectively. Figure 10.3 examined the number of explicit changes on each profile driven by the assumption that users avoid editing their profiles very often when these reflect an accurate state of their interests and facets. Data along this direction illustrated that the number of explicit changes to user profiles is kept to a minimum, while they exhibit a decreasing trend, converging to zero towards the end of the evaluation study.

Based on these observations, it is safe to conclude that the current implementation of the profile learning scheme is able to recognize the different facets of user needs and update their state in Colombus effectively.

1.b Adaptation improves search effectiveness

Measuring the ability to adapt to user profiles has never been an easy task. Although data collected through the lab-based experiments could be employed along this direction, their scope was purposely limited to tuning Colombus. Background logging can be employed to measure the adaptivity of user profiles by again looking at the number of explicit changes presented in figure 10.3. Similarly, measuring the degree of adaptivity through questionnaires is not always possible, since experimental subjects are often unable to detect their own drifts to new interests. Also, the duration of this evaluation period is rather short to measure the system behavior in handling drifting user concepts.

I have attempted to capture the effect of adaptive profile learning in figure 10.5. At the midpoint of the user study the adaptivity was rated with a positive score of 3.25 points, while there has been a significant increase to 3.78 points after 14 evaluation days. Also, the trend shows that as experimental subjects use the system for their everyday searching tasks, the effectiveness of the profile learning scheme increases

analogously.

2.a Combination of implicit and explicit feedback can model evolving user needs

Taking into account the results obtained and demonstrated in figures 10.3 and 10.4, Colombus can model user needs in an accurate and effective way. According to the data presented in figure 10.3, the number of explicit modifications to user profiles exhibit a decreasing trend throughout the evaluation study, converging to 0 towards the last days. Also, I recorded less than a change every two days on average for each of the three categories (add, remove, amend interest). Similarly, questionnaire data collected and presented as part of figure 10.4 indicate that experimental subjects reduce the explicit changes in their profile over time. The data included in these figures provide adequate evidence that profiling users' needs through Colombus is being done in an effective and successful way. This evidence supports this part of our hypothesis.

2.b Users avoid editing their profiles very often

Based on logging data recorded throughout the user experiments, we have observed that profile changes, in the sense of interest additions, deletions and amendments, have been maintained at very low levels. The results along this direction were presented in figure 10.3, which illustrates that Colombus' adaptive profile learning scheme works very effectively. Also, taking into account that the system builds user profiles solely based on implicit click-through data, the learning model is simple and easy to understand, yet efficient and effective.

In a similar fashion, figure 10.4 is based on questionnaire data captured at regular evaluation intervals. These results illustrate that the age of user profiles are directly related to their accuracy. Profiles measured at the end of the evaluation period appear to be much closer to user needs compared to the same ones a few days earlier. Thus, it is safe to conclude that users avoid editing their profiles very often.

3.a It can fetch additional relevant documents

The claim above ascertains to what degree users perceive additional documents, retrieved in their personal page, to be relevant, an important factor in any information retrieval system. By investigating the accuracy of document recommendations throughout the user study we can gain further insight in this aspect.

As figure 10.6 illustrates, participants found document recommendations as very relevant (4 out of 5) and liked (4.57 out of 5). Also, all measures in the underlying table shoe a constant increase between the first and second time intervals. In essence, as experimental subjects started getting more familiar with the system, they started getting more value out of their personalized home page. page contains even more relevant documents.

The access patterns and statistics of document recommendations constitutes another important source of information. Figure 10.9 illustrates that document recommendation access shows a significant increase over the evaluation period. Based on this observation, experimental subjects increasingly recognized and appreciated the value of document recommendations during the duration of this study.

Also, discovering documents on the web relevant to a user's profile is not always adequate. Several other factors have to be taken into account before concluding whether a information portal like Colombus is successful, such as the novelty of recommended documents. Along this direction, I have gathered and analyzed statistics from questionnaire data earlier presented in figure 10.8. Almost 80% of experimental subjects declared that they were able to find at least one document relevant to their needs that they hadn't seen in the past. Also, in the same figure it is shown that document recommendations in general were perceived as very interesting.

3.b Presenting information on a priority basis is appropriate and effective

As illustrated on figure 5.4, additional documents are presented on a priority basis, depending on the access pattern of the user's interests. Presenting these on a priority basis relies on the assumption that users will initially seek additional documents regarding their most popular interests. Even though the evaluation data are inadequate, to decide whether or not this part of the hypothesis holds, qualitative data collected through questionnaires suggest that the system's functionality and usability is not compromised by this design decision.

10.5 User Comments

Apart from the quantitative measures mentioned in previous sections, I also gathered qualitative results from open-ended questions. Comments such as

- "It provided me with all the relevant documents I needed for my search"
- "It saved me time having to find things in the web as the search was already done on my behalf"

illustrate that most users perceived it as a great assistant for their searching needs. The system's ability to model adaptive user needs and implicitly create their search profiles, through the profile learning scheme, was judged as "*great*", while the profile adaptivity was found "*quite a magic feature*". Finally, it was confirmed that it "*saves a lot of time*", because users "*can find additional information automatically*".

Part V

Conclusion

Part IV described the evaluation of Colombus consisting of two main stages: system calibration and performance evaluation. I defined a large scale lab-based simulation methodology to discover an optimal system configuration based around the concept of searcher simulations. Then, I also designed and carried out a task-based user-centered evaluation approach to assess the performance of Colombus against a series of measures as well as SearchIT, a retrieval system unaware for user profiles. The results showed that Colombus can adapt rapidly to concept changes and assist users by providing a range of accurate document recommendations. In this part, I conclude this thesis and present possible ways to expand this work in the future drawn from findings obtained in the evaluation experiments.

Chapter 11

Conclusion

In this thesis I focused on engineering a recommendation assistant based on the use of implicit feedback gathering to aid users in their everyday searching needs. A range of term weighting models were integrated and assessed during an attempt to calibrate the system. A vast amount of effort was put towards finding ways to detect and adapt to concept drifts to keep up-to-date with changing user needs. Along Part I, I described the current state of research in the field of information filtering systems and set the grounds for exploration. Part II described the dynamics of user search needs by identifying the difference between short and long term needs, as well as strategies with respect to profile representation. Furthermore, a range of different approaches for adapting to concept drifts was presented and analyzed. Parts I and II motivate the development of Colombus, which were presented along Part III focusing on the relevance feedback and profiling algorithms, as well as the work involved in the user interface. Last but not least, along Part IV I proposed a hybrid evaluation methodology for recommendation systems based on a combination of simulation-based experiments and a user-centered study. The remainder of this chapter investigates and summarizes the main findings and contributions of this thesis.

11.1 Implicit Feedback

Throughout 2.1 I investigated explicit and implicit feedback gathering mechanisms from a research point of view and presented the benefits and shortcomings of each approach. Taking into account that one of my main objectives was to relieve the burden from users imposed throughout the document assessment phase, explicit feedback was found unsuitable for our needs.

On the other hand, implicit feedback gathering systems infer document ratings by monitoring searcher interaction with the user interface. Depending on the underlying relevance feedback collection policy, each system monitors different sources and derives a document rating. Although this approach has gathered much attention especially over the past few, very few recommendation assistants have been fully engineered. Our main objective was to build a system that could be used with no maintenance and effort on behalf of the searchers. Along this aspect, Colombus is equipped with an implicit feedback collection policy based on click-through data.

Also, as detailed along section 6.4, click-through data are collected by monitoring user interaction with the document surrogate. Following this approach, I also implemented a series of weighting schemes ranging from a basic binary model to complex voting-based statistical techniques like Jeffrey's conditioning. In fact, voting-based schemes treated click-through surrogates in a different way by boosting or reducing weights based on the confidence of the underlying component. A request to view the document summary indicates less confidence than a request to bookmark. To our best knowledge there is very little work on the area of voting weighting schemes combined with implicit relevance feedback gathering. From an evaluation point of view, we discovered that a voting version of the term frequency model results in the best overall performance. Along this direction, not only we investigated a new family of weighting schemes in real-world conditions, but also discovered avenues for improving the performance of document recommendations by approximating the searchers' interest indicated through interaction with the document surrogates.

11.2 Adaptive Document Recommendations

The dynamic nature of search interests was covered in section 3.1. As users gather more information on some particular topic, they are likely to change their notion of relevance resulting to a concept drift. Similarly, a sudden change in the user's search priorities is defined as a concept shift. The rate of change is directly related to the required rate of learning of the profile learning approach: A slow but constant change requires a large window of data, while a rapid change should quickly invalidate older data.

Although the notion of drifting concepts has been investigated by several researchers over the past years, most studies have defined the theoretical framework. Others have developed adaptive information filtering systems, but the implications of document recommenders along this direction are much more complex than other profile-aware alternatives. The effect of maladaptive user profiles becomes apparent shortly after the recommendation process. While the user expects assistance related to his latest user needs, the document recommendations produced do not reflect his conceptual profile.

Section 3.2 has covered a range of different techniques to detect and adapt to concept changes focusing on the design behind time windows and decay functions. A decay function is a way to weight previously seen information based on their age. Such a function has been developed and integrated as part of Colombus adaptation strategy. Since the adaptive policy assumes constantly changing user profile, potential avenues of future work could be on the development of a dynamic adaptation strategy that detects the current rate of change and adjusts the profile accordingly. Also, in terms of the half-life of the adaptation policy, a proper evaluation could be carried out in calibrating this particular setting.

11.3 Evaluation

The evaluation of Colombus was approached using a novel technique combining both searcher-simulation based experiments and studies conducted in realistic conditions with real users. At the first stage, the system was evaluated using a series of searcher stereotypes with simulated characteristics such as patience and topic experience. These lab-based experiments were executed in large batches while I experimented with different system settings, from threshold configuration to weighting schemes. The aim was to derive

an optimal configuration that works well for a range of different searcher stereotypes. Along this direction, Colombus was evaluated against three scenarios each focusing on different aspects. Apart from the first scenario that assessed the effectiveness of document recommendations as a standalone component, the other two scenarios focused on the profile learning aspects such as the adaptivity of user profiles, as well as their evolution throughout the search episodes. In order to gather more information regarding the latter, I introduced and employed the Profile Evolution Trend (PET) measure that indicates how each profile evolved over time. The results of this simulation stage highlighted that a voting-based term frequency (TF) scheme combined with average threshold settings outperforms the remaining configurations on the majority of simulation scenarios.

The second stage of Colombus evaluation involved 15 searchers using the system for over 14 days, while I monitored their interactions using both quantitative and qualitative means. Users were asked to complete a set of questionnaires to capture their views throughout the evaluation stage, but background logging was also used heavily. Furthermore, a second system, called SearchIT, was employed as a baseline in order to compare such a non-adaptive search tool against Colombus. The results showed that the majority of experimental subjects thought that they would benefit from an adaptive commercial search tool, while the usage of Colombus against SearchIT increased on a daily basis indicating the users' implicit preference.

Chapter 12

Future Work

In the following sections I outline a number of areas for possible future work. These areas either describe aspects of this project that are worth of further investigation, or that can be improved and enhanced.

12.1 Adaptive User Needs

As detailed in section 10.4, evaluating the adaptivity of profile learning schemes is never an easy task. However, a special user evaluation tailored solely for measuring Colombus reaction to drifting concepts could be employed in the future to gain further insight along this aspect. Such a methodology would be designed to inject drifts to experimental tasks to force participants to follow certain paths. The objective is to start from general task and modify the task description at several time intervals during the user study to make it more focused. Thus, users will be forced to alter their searches in order to gather information related to the new task description. This way we could create a simulated concept drift to evaluate the degree of adaptivity during profile management scheme. Table12.1 presents a sample experimental topic and a possible evolution scenario (changes in italic. This scenario could be attempted in a future evaluation of Colombus.

Evaluation Days	Task Description
Day 1-5	You recently graduated from university and are looking to start a job in the industry. As you don't have any previous experience in this aspect, you are looking to collect more information with regards to the interview process. Use the information retrieval tool provided to find more information with regards to this topic.
Day 6-10	You recently graduated from university *with degree in software engineering* and are looking to start a job in the industry. As you don't have any previous experience in this aspect, you are looking to collect more information with regards to the interview process. Use the information retrieval tool provided to find more information with regards to this topic.
Day 11-15	You recently graduated from university *with degree in software engineering* and are looking to start a job in the industry. *As you have worked before in a games development company, you are looking to work in a similar field.* Use the information tool provided to find more information with regards to this topic.
Day 16-20	You recently graduated from university *with degree in software engineering* and are looking to start a job in the industry. *As you have worked before in a games development company specialized in driving simulators, you are looking to work in a similar field.* Use the information tool provided to find more information with regards to this topic.

Table 12.1: The evolution of an experimental task

12.2 The Synonym Problem

During the simulation study, I observed a problem related to the addition of new information in user profiles. The *"synonym effect",* as it has been named, occurs especially on vector-based user interests and profiles that have been recently created and thus have very few information. This issue was described in detail in section 8.4.

Although the synonym issue hasn't been properly analysed, the data we collected throughout the simulation study is adequate to suggest a few possible solutions around it. As a first attempt we could try

increasing the number of terms contained in each term set gathered during a search iteration. Then, the synonym terms are more likely to be contained in the term set and the chances of getting the ”synonym effect” are slimmer. However, the increase in the term set size is paired with an analogous increase in the dimensions of the profile, which could lead into problems in the future.

A more sophisticated approach would be to use a Word-net base dictionary to expand the initial relevance set with the synonyms of each word. For instance, the term *”car”* could be expanded to *”vehicle”, ”automobile”, ”motor”, ”roadster”, ”wheel”* each assigned with a weight depending to its relevance with the initial term. Therefore, not only the original informational term is maintained, but a new set of terms possibly relevant to the subject are added for further processing. Such word-net dictionaries are widely available on the web for development use, but again a possible dimensionality problem is lurking.

12.3 Sources of Document Recommendations

Section 6.4 refers to the implicit feedback gathering sources used to extract information on user search needs. Currently, the system formulates queries based on the keywords in a the user profile and automatically issues new searches to locate additional information. Another way to improve the performance and effectiveness of this information retrieval tool is to invoke to crawl a large amount of topic-specific portals and extract information on a certain aspect. A focused crawler seeks and indexes pages on a specific set of topics that represent a relatively narrow segment of the web. Thus, a focused crawler can improve the recommendation algorithm by discovering a set of relevant documents, unable to be discovered with a simple search iteration. The scope of this project is beyond the development of such a tool, but there is certainly future work in this aspect.

Bibliography

Allan, J. (1996), Incremental relevance feedback for information filtering, *in* 'SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 270–278.

Allan, J., Carbonell, J., Doddington, G., Yamron, J. & Yang, Y. (1998), Topic detection and tracking pilot study: Final report, *in* 'Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop', pp. 194–218.

Allan, J., Harding, S., Fisher, D., Bolivar, A., Guzman-Lara, S. & Amstutz, P. (2005), Taking teeeopic detection from evaluation to practice, *in* 'HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 4', p. 101.1.

Arampatzis, A., Beney, J., Koster, C. & Van der Weide, T. (2000), Incrementality, half-life, and threshold optimization, for adaptive document filtering, *in* 'Proceedings of the 9th text retrieval conference', pp. 589–600.

Arampatzis, A. & Van der Weide, T. (2001), Document filtering as an adaptive and temporally-dependent process, *in* 'Proceedings of the BCS-IRSG European Colloquium on IR Research'.

Arampatzis, A., Van der Weide, T., Koster, C. & van Bommel, P. (2000), Term selection for filtering based on distribution of terms over time, *in* 'Proceedings of RIAO'2000 Content-Based Multimedia Information Access', pp. 1221–1237.

Armstrong, R., Freitag, D., Joachims, T. & Mitchell, T. (1995), Webwatcher: A learning apprentice for the world wide web, *in* 'AAAI 1995 Spring Symp. Information Gathering from Heterogeneous, Distributed Environments', pp. 6–12.

Asnicar, F. & Tasso, C. (1997), ifweb: A prototype of user model-based intelligent agent for information filtering and navigation in the world wide web, *in* 'Proceedings of the 6th International Conference on User Modelling'.

Baeza-Yates, R., Hurtado, C. & Mendoza, M. (2004), Query recommendation using query logs in search engines, *in* 'Proceedings of the International Workshop on Clustering Information over the Web', pp. 588–596.

Baeza-Yates, R. & Ribeiro-Neto, B. (1999), *Modern Information Retrieval*.

Balabanovic, M. (1997), An adaptive web page recommendation service, *in* 'AGENTS '97: Proceedings of the first international conference on Autonomous agents', pp. 378–385.

Bauer, E. & Kohavi, R. (1999), 'An empirical comparison of voting classification algorithms: Bagging, boosting, and variants', *Machine Learning* **36**(1-2), 105–139.

Beitzel, M., Jensen, C., Chowdhury, A., Grossman, D. & Frieder, O. (2004), Hourly analysis of a very large topically categorized web query log, *in* 'SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 321–328.

Belkin, N. (1997), User modeling in information retrieval, *in* 'Sixth International Conference on User Modeling (Tutorial)'.

Belkin, N. (2000), 'Helping people find what they don't know', *Communications of the ACM* **43**(8), 58–61.

Belkin, N. & Croft, W. (1992), 'Information filtering and information retrieval: two sides of the same coin?', *Communications of the ACM* **35**(12), 29–38.

Borlund, P. (2000), 'Experimental components for the evaluation of interactive information retrieval systems', *Journal of Documentation* **56**, 71–90.

Borlund, P. & Ingwersen, P. (1997), 'The development of a method for the evaluation of interactive information retrieval systems', *Journal of Documentation* **53**(3), 225–250.

Brin, S. & Page, L. (1998), The anatomy of a large-scale hypertextual web search engine, *in* 'WWW7: Proceedings of the seventh international conference on World Wide Web 7', pp. 107–117.

Carpineto, C., De Mori, R. & Romano, G. (1998), Informative term selection for automatic query expansion, *in* 'The 7th Text REtrieval Conference', pp. 363–369.

Carreira, R., Crato, J., Gonçalves, D. & Jorge, J. (2004), Evaluating adaptive user profiles for news classification, *in* 'IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces', pp. 206–212.

Chen, H. & Dumais, S. (2000), Bringing order to the web: automatically categorizing search results, *in* 'CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems', pp. 145–152.

Chen, L. & Sycara, K. (1998), Webmate: a personal agent for browsing and searching, *in* 'AGENTS '98: Proceedings of the second international conference on Autonomous agents', pp. 132–139.

Chien-Kang, H., Lee-Feng, C. & Yen-Jen, O. (2003), 'Relevant term suggestion in interactive web search based on contextual information in query session logs', *Journal of the American Society for Information Science and Technology* **54**(7), 638–649.

Chirita, P. A., Olmedilla, D. & Nejdl, W. (2004), Pros: A personalized ranking platform for web search, *in* 'Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2004)', number 3137, pp. 34–43.

Cui, H., Wen, J. R., Nie, J. Y. & Ma, W. Y. (2003), 'Query expansion by mining user logs', *IEEE Transaction on Knowledge and Data Engineering* **15**(4), 829–839.

Dennis, S., McArthur, R. & Bruza, P. (1998), Searching the world wide web made easy? the cognitive load imposed by query refinement mechanisms, *in* 'Proceedings of the Third Australian Document Computing Symposium (ADCS'98)', pp. 65–71.

Dietterich, T. G. (2000), 'Ensemble methods in machine learning', *Lecture Notes in Computer Science* **1857**, 1–15.

Efthimiadis, E. (2000), 'Interactive query expansion: a user-based evaluation in a relevance feedback environment', *Journal of the American Society for Information Science and Technology* **51**(11), 989–1003.

Godoy, D. & Amandi, A. (2005), 'User profiling in personal information agents: a survey', *The Knowledge Engineering Review* **20**(4), 329–361.

Harman, D. (1988), Towards interactive query expansion, *in* 'SIGIR '88: Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 321–331.

Harman, D. (1992*a*), 'Evaluation issues in information retrieval', *Information Processing and Management: an International Journal* **28**(4), 439–440.

Harman, D. (1992*b*), Relevance feedback revisited, *in* 'SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 1–10.

Haveliwala, T. H. (2003), 'Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search', *IEEE Transactions on Knowledge and Data Engineering* **15**(4), 784–796.

Hearst, M. & Pedersen, J. O. (1996), Reexamining the cluster hypothesis: scatter/gather on retrieval results, *in* 'SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 76–84.

Herlocker, J. L., Konstan, J. A., Terveen, L. G. & Riedl, J. T. (2004), 'Evaluating collaborative filtering recommender systems', *ACM Transactions on Information Systems (TOIS)* **22**(1), 5–53.

Hersh, W. & Hickam, D. (1995), 'An evaluation of interactive boolean and natural language searching with an online medical textbook', *Journal of the American Society for Information Science* **46**(7), 478–489.

Hoscher, C. & Strube, G. (2000), Web search behavior of internet experts and newbies, *in* 'Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking', Vol. 33, pp. 337–346.

Hull, D. A. (1997), The trec-6 filtering track: Description and analysis, *in* 'Proceedings of the Sixth Text Retrieval Conference (TREC-6)', pp. 45–68.

Ingwersen, P. & Willett, P. (1995), 'An introduction to algorithmic and cognitive approaches for information retrieval', *LIBRI* **45**(3), 160–177.

Jansen, B. J. & Spink, A. (2005), 'An analysis of web searching by european alltheweb.com users', *Information Processing and Management: an International Journal* **41**(2), 361–381.

Jansen, B. & Pooch, U. (2001), 'A review of web searching studies and a framework for future research', *Journal of the American Society for Information Science and Technology* **52**(3), 235–246.

Jansen, B., Spink, A., Bateman, J. & Saracevic, T. (1998), 'Real life information retrieval: a study of user queries on the web', *SIGIR Forum* **32**(1), 5–17.

Jeh, G. & Widom, J. (2003), Scaling personalized web search, *in* 'WWW '03: Proceedings of the 12th international conference on World Wide Web', pp. 271–279.

Joachims, T. (1997), A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, *in* D. H. Fisher, ed., 'Proceedings of ICML-97, 14th International Conference on Machine Learning', pp. 143–151.

Joachims, T. (2002), Optimizing search engines using clickthrough data, *in* 'KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining', pp. 133–142.

Joachims, T., Freitag, D. & Mitchell, T. (1995), Web watcher: A tour guide for the world wide web, *in* 'AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments', pp. 6–12.

Joho, H. & Jose, J. (2008), 'Effectiveness of additional representations for the search result presentation on the web', *Information Processing and Management: an International Journal* **44**(1), 226–241.

Kamba, A. T. & Bharat, K. (1995), The Krakatoa Chronicle: An interactive personalized newspaper on the Web, *in* 'In Proceedings of the Fourth International World Wide Web Conference', pp. 159–170.

Keenoy, K. & Levene, M. (2005), Personalisation of web search, *in* 'Intelligent Techniques for Web Personalization', pp. 201–228.

Kelly, D. & Belkin, N. (2001), Reading time, scrolling and interaction: exploring implicit sources of user preferences for relevance feedback, *in* 'SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 408–409.

Kelly, D., Diaz, F., Belkin, N. & Allan, J. (2004), A user-centered approach to evaluating topic models, *in* 'Proceedings of the European Conference on Information Retrieval (ECIR '04)', pp. 27–41.

Kelly, D. & Teevan, J. (2003), 'Implicit feedback for inferring user preference: a bibliography', *SIGIR Forum* **37**(2), 18–28.

Khopkar, Y., Spink, A., Giles, C. L., Shah, P. & Debnath, S. (2003), 'Search engine personalization: An exploratory study', *First Monday* **8**(7).

Kim, J., Oard, D. & Romanik, K. (2000), Using implicit feedback for user modeling in internet and intranet searching, Technical report, Technical Report, College of Library and Information Services, University of Maryland at College Park.

Klinkenberg, R. (2001), Using labeled and unlabeled data to learn drifting concepts, *in* 'Proceedings of International Joint Conference of Artificial Intelligence (IJCAI-01) Workshop Learning from Temporal and Spatial Data', pp. 16–24.

Klinkenberg, R. (2004), 'Learning drifting concepts: Example selection vs. example weighting', *Intelligent Data Analysis* **8**(3), 281–300.

Klinkenberg, R. & Renz, I. (1998), Adaptive information filtering: Learning in the presence of concept drifts, *in* 'Proceedings of AAAI Workshop Learning for Text Categorization', pp. 33–40.

Klinkenberg, R. & Ruping, S. (2002), 'Concept drift and the importance of examples', *In Franke, Jürgen and Nakhaeizadeh, Gholamreza and Renz, Ingrid (editors), Theoretical Aspects and Applications* pp. 55–77.

Kolter, J. Z. & Maloof, M. A. (2003), Dynamic weighted majority: A new ensemble method for tracking concept drift, *in* 'ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining', p. 123.

Konstan, J. A. & Riedl, J. (1999), Research resources for recommender systems, *in* 'Proceedings of CHI' 99 Workshop Interacting with Recommender Systems'.

Koychev, I. (2001), Learning about user in the presence of hidden context, *in* 'Proceedings of the UM2001 Workshop on Machine Learning for User Modeling', pp. 69–101.

Koychev, I. & Schwab, I. (2000), Adaptation to drifting user's interests, *in* 'Proceedings of ECML2000 Workshop: Machine Learning in New Information Age', pp. 39–45.

Kullback, S. & Leibler, R. (1951), 'On information and sufficiency', *The Annals of Mathematical Statistics* **22**(1), 49–86.

Kumaran, G. & Allan, J. (2006), Eliciting information for adaptive retrieval, *in* 'Proceedings of the First International Workshop on Adaptive Information Retrieval (AIR 2006)', pp. 18–19.

Lam, W., Mukhopadhyay, S., Mostafa, J. & Palakal, M. (1996), Detection of shifts in user interests for personalized information filtering, *in* 'SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 317–325.

Lanquillon, C. & Renz, I. (1999), Adaptive information filtering: detecting changes in text streams, *in* 'CIKM '99: Proceedings of the eighth international conference on Information and knowledge management', pp. 538–544.

Lieberman, H. (1995), Letizia: An agent that assists web browsing, *in* 'Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)', pp. 924–929.

Liu, J., Wong, C. K. & Hui, K. (2003), 'An adaptive user interface based on personalized learning', *IEEE Intelligent Systems* **18**(2), 52–57.

Manber, U., Patel, A. & Robison, J. (2000), 'Experience with personalization of yahoo!', *Communications of the ACM archive* **43**(8), 35–39.

Meng, X. (2006), A comparative study of performance measures for information retrieval systems, *in* 'ITNG '06: Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)', pp. 578–579.

Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A. & Riedl, J. (2003), Movielens unplugged: Experiences with an occasionally connected recommender system, *in* 'Proceedings of ACM 2003 Conference on Intelligent User Interfaces (IUI'03) (Accepted Poster)'.

Mladenic, D. (1996), Personal webwatcher: Design and implementation, Technical report, Technical Report IJS-DP-7472, School of Computer Science, Carnegie-Mellon University, Pittsburgh, USA, October.

Morita, M. & Shinoda, Y. (1994), Information filtering based on user behavior analysis and best match text retrieval, *in* 'Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 272–281.

Mostafa, J., Mukhopadhyay, S. & Palakal, M. (2003), 'Simulation studies of different dimensions of users' interests and their impact on user modeling and information filtering', *Information Retrieval* **6**(2), 199–223.

Nelson, M. R. (1994), 'We have the information you want, but getting it will cost you!: held hostage by information overload.', *Crossroads* **1**(1), 11–15.

Nichols, D. (1997), Implicit rating and filtering, *in* 'Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering', pp. 31–36.

Nichols, D., Twindale, M. & Paice, C. (1997), Recommendation and usage in the digital library, Technical report, Computing Department, Lancaster University.

Nick Street, W. & YongSeog, K. (2001), A streaming ensemble algorithm (sea) for large-scale classification, *in* 'KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining', pp. 377–382.

Oard, D. & Kim, J. (2001), Modeling information content using observable behavior, *in* 'Proceedings of the 64th Annual Meeting of the American Society for Information Science and Technology', Vol. 38, pp. 481–488.

Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C. & Johnson, D. (2005), Terrier information retrieval platform, *in* 'Proceedings of the 27th European Conference on IR Research (ECIR 2005)', Vol. 3408, pp. 517–519.

Parsons, J., Ralph, P. & Gallagher, K. (2004), Using viewing time to infer user preference in recommender systems, *in* 'Proceedings of the AAAI Workshop in Semantic Web Personalization'.

Payne, T. R., Edwards, P. & Green, C. L. (1997), 'Experience with rule induction and k-nearest neighbor methods for interface agents that learn', *IEEE Transactions on Knowledge and Data Engineering* **9**(2), 329–335.

Pazzani, M. J., Muramatsu, J. & Billsus, D. (1996), Syskill webert: Identifying interesting web sites, *in* 'Proceedings of the 13th National Conference on Artificial Intelligence', Vol. 1, pp. 54–61.

Psarras, I. & Jose, J. (2006), A system for adaptive information retrieval, *in* 'In Proceedings of Adaptive Hypermedia conference', pp. 313–317.

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P. & Riedl, J. (1994), Grouplens: An open architecture for collaborative filtering of netnews, *in* 'CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work', pp. 175–186.

Robertson, S. (1990), 'On term selection for query expansion', *Journal of Documentation* **46**(4), 359–364.

Rocchio, J. J. (1971), *Relevance feedback in information retrieval*, Vol. Chapter 14.

Ruthven, I. (2003), Re-examining the potential effectiveness of interactive query expansion, *in* 'SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval', pp. 213–220.

Salton, G. (1989), *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*.

Silverstein, C., Henzinger, M., Marais, H. & Moricz, M. (1998), Analysis of a very large altavista query log, Technical Report 1998-014, Digital SRC.

Stanley, K. (2001), Learning concept drift with a committee of decision trees, Technical report, Technical Report UT-AI-TR-03-302, Computer Sciences Department, University of Texas.

Stefani, A. & Strappavara, C. (1998), Personalizing access to web sites: The siteif project, *in* 'In Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98', pp. 20–24.

Stevens, F. (1993), Knowledge-based assistance for accessing large, poorly structured information spaces, PhD thesis.

White, R., Jose, J. & Ruthven, I. (2003), 'A task-oriented study on the influencing effects of query-biased summarisation in web searching', *Information Processing and Management: an International Journal* **39**(5), 707–733.

White, R., Jose, J., Van Rijsbergen, C. & Ruthven, I. (2004), 'A simulated study of implicit feedback models', *In Proceedings of the 26th European Conference on Information Retrieval Research (ECIR '04)* pp. 311–326.

White, R. & Marchionini, G. (2007), 'Examining the effectiveness of real-time query expansion', *Information Processing and Management: an International Journal* **43**(3), 685–704.

White, R., Ruthven, I. & Jose, J. (2002*a*), Finding relevant documents using top ranking sentences: an evaluation of two alternative schemes, *in* 'SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval', pp. 57–64.

White, R., Ruthven, I. & Jose, J. (2002*b*), The use of implicit evidence for relevance feedback in web retrieval, *in* 'Proceedings of the 24th BCS-IRSG European Colloquium on IR Research', pp. 93–109.

White, R., Ruthven, I., Jose, J. & Van Rijsbergen, C. (2005), 'Evaluating implicit feedback models using searcher simulations', *ACM Transactions on Information Systems (TOIS)* **23**(3), 325–361.

Widmer, G. & Kubat, M. (1996), 'Learning in the presence of concept drift and hidden contexts', *Machine Learning* **23**, 69–101.

Xue, G., Zeng, H., Chen, Z., Yu, Y., Ma, W., Xi, W. & Fxan, W. (2004), Optimizing web search using web click-through data, *in* 'CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management', pp. 118–126.

Zhu, T., Greiner, R. & Haubl, G. (2003), An effective complete-web recommender system, *in* 'Proceedings of the 12th International World Wide Web Conference (WWW2003)'.

Appendix A

Simulation Study

A.1 Sample Simulation Run Record

Please not that for simplicity, only one iteration log is shown. The format is *index of iteration*, *recommendation precision*, *correlation between profile and extracted terms*, *correlation between profile and query*, *PET measure.*

0,0.5,0.18555550934404702,0.18555550433830698,1

0,0.5,0.2440755763243709,0.250542246145053,1

0,0.5,0.26802689870084023,0.263512445647376,1

0,0.5,0.2741010660874029,0.2680974477432117,1

0,0.5,0.2769792278717036,0.27044681569372025,1

0,0.5,0.2786374540622147,0.27187221534834716,1

0,0.5,0.2797097783849571,0.27282128201294803,1

0,0.5,0.2804580241989093,0.27349205468045273,1

0,0.5,0.2810088917290608,0.2739872066440319,1

0,0.5,0.28143094128279217,0.27436546873884127,1

1,0.0,0.0018548891827160322,0.0018548890566479153,1

1,0.0,0.0015231384807655047,0.00172352535007335,1

1,0.0,0.0015231384807655047,0.0016875078158553704,1

1,0.0,0.0015231384807655047,0.001672662656856012,1

1,0.0,0.0015231384807655047,0.0016655602005943514,1

1,0.0,0.0015231384807655047,0.0016619205346267672,1

1,0.0,0.0015231384807655047,0.0016599897782405228,1

1,0.0,0.0015231384807655047,0.0016589466899064186,1

1,0.0,0.0015231384807655047,0.0016583777028955937,1

1,0.0,0.0015231384807655047,0.0016580657128675437,1

Appendix B

Evaluation Questionnaires

B.1 Entry Questionnaire

Entry Questionnaire

UNIVERSITY
of
GLASGOW

This questionnaire will provide us with background information that will help us analyse the answers you give in later stages of this experiment. Feel free to skip a question if you feel it is too personal.

- Your age: _____
- Your sex (Please circle): M / F
- Occupation: _____
- Level of Education: _____
- Degree : _____ Year: _____
- How would you rate your computer and internet knowledge?

Not good at all				Very Good
1	2	3	4	5

- Overall, for how many years have you been doing online searching? _____

- How often do you carry out online searching at home or work? Please circle the closest option.

Rarely	1-2 times a month	1-2 times a week	Once a day	Several times a day

- How often do you find what you are looking for when using search engines?

Not often at all				Very often
1	2	3	4	5

- Favourite Search Engine : _____

- Using your favourite search engine is generally:

	Very	Reasonably	Neither/Nor	Reasonably	Very	
Stressful	1	2	3	4	5	Relaxing
Complex	1	2	3	4	5	Simple
Frustrating	1	2	3	4	5	Satisfying

- What features would you like to have in your favourite search engine (circle as many as you want)?

(a) Adaptive retrieval (b) Document recommendations (c) Search history (d) Query assistance
(e) Other: _____

B.2 Post Search Questionnaire

Post search Questionnaire (Interface 1)

UNIVERSITY *of* GLASGOW

To evaluate the system you have been using, we now ask you to answer some questions about it. Take into account that we are interested in knowing your opinion: answer questions freely, and consider there are no right or wrong answers.

Please remember that we are evaluating the system, not your performance.

1. System's Performance

- The additional relevant documents retrieved by the system were:

Not Relevant				Relevant
1	2	3	4	5

Inadequate				Adequate
1	2	3	4	5

- How would you rate the adaptivity of the system (i.e. tracking and accommodating changes in your search concepts):

Not Adaptive				Very Adaptive
1	2	3	4	5

- How often did you have to edit your profile to improve the system's performance?

Not often at all				Very Often
1	2	3	4	5

- How would you rate the system in capturing you temporal needs

Not good at all				Very Good
1	2	3	4	5

- How would you rate the system in profiling your interests in general.

Not good at all				Very Good
1	2	3	4	5

- What did you like most about this system?

- What did you hate most about this system?

- Please write any other comments about this system.

2. User Interface

- How would you rate the user interface :

	Very	Reasonably	Neither/Nor	Reasonably	Very	
Difficult to use	1	2	3	4	5	Easy to use
Difficult to learn	1	2	3	4	5	Easy to learn
Difficult to understand	1	2	3	4	5	Easy to understand

- What did you hate most about the user interface?

- What did you like most about the user interface?

- Please write any comments regarding the user interface of the system:

B.3 Exit Questionnaire

Exit Questionnaire

The aim of this experiment was to investigate the effectiveness of a personalized information retrieval system. Please consider the entire experience that you just had when you respond to the following questions.

UNIVERSITY *of* GLASGOW

- How many different tasks did you search for: _____

- The tasks we asked you to perform was:

	Very	Reasonably	Neither/Nor	Reasonably	Very	
Clear	1	2	3	4	5	Unclear
Simple	1	2	3	4	5	Complex
Interesting	1	2	3	4	5	Uninteresting
Relevant to you	1	2	3	4	5	Not relevant to you

- How would you rate Colombo's performance in general?

Not good at all				Very Good
1	2	3	4	5

- How useful do you think Colombo was in retrieving relevant documents on your behalf?

Not good at all				Very Good
1	2	3	4	5

- How do you think Colombo compares to your favourite search engine.

Colombo is worse				Colombo is better
1	2	3	4	5

- Please write any comments about the system or experiment. Did you find anything particularly useful/interesting? Was there anything annoying in the system/experiment?

B.4 Comparison Questionnaire

System Comparison

To evaluate the systems you have been using, we now ask you to answer some questions about it. Take into account that we are interested in knowing your opinion: answer questions freely, and consider there are no right or wrong answers. Please remember that we are evaluating the systems, not your performance.

- How effective do you think the system is:

System 1 – Colombo

Very Effective				Not Effective
1	2	3	4	5

System 2 – SearchIt

Very Effective				Not Effective
1	2	3	4	5

- How much did the system help you to find relevant documents?

System 1 – Colombo

A lot				Not at all
1	2	3	4	5

System 2 – SearchIt

A lot				Not at all
1	2	3	4	5

- What was your personal preference between the two systems?

  System 1 – Colombo

  System 2 – SearchIt

- Please write any additional comments regarding the two systems you used

Appendix C

Profile Structure

C.1 Sample XML Profile

```
<PROFILE>
  <INTEREST>
    <NAME>house inflation</NAME>
    <LAST>1201808323016</LAST>
    <ACCESSED>5</ACCESSED>
    <ID>0</ID>
    <ITERATION>
       <TIME>1201293208265</TIME>
       <TERM>
          <WORD>agent</WORD>
          <FREQ>1.0</FREQ>
       </TERM>
       <TERM>
          <WORD>bed</WORD>
```

<FREQ>1.0</FREQ>
</TERM>
<TERM>
<WORD>bedroom</WORD>
<FREQ>1.0</FREQ>
</TERM>
<TERM>
<WORD>buy</WORD>
<FREQ>1.0</FREQ>
</TERM>
<TERM>
<WORD>house</WORD>
<FREQ>2.0</FREQ>
</TERM>
<TERM>
<WORD>inflation</WORD>
<FREQ>2.0</FREQ>
<TERM>
<WORD>uk</WORD>
<FREQ>2.0</FREQ>
</TERM>
</ITERATION>
<ITERATION>
<TIME>1201808323015</TIME>
<TERM>
<WORD>buyers</WORD>
<FREQ>31.0</FREQ>
</TERM>

```
   <TERM>
      <WORD>excel</WORD>
      <FREQ>59.0</FREQ>
   </TERM>
   <TERM>
      <WORD>house</WORD>
      <FREQ>60.0</FREQ>
   </TERM>
   <TERM>
      <WORD>kb</WORD>
      <FREQ>59.0</FREQ>
   </TERM>
   <TERM>
      <WORD>prices</WORD>
      <FREQ>58.0</FREQ>
   </TERM>
   <TERM>
      <WORD>tablems</WORD>
      <FREQ>57.0</FREQ>
   </TERM>
   <TERM>
      <WORD>unit</WORD>
      <FREQ>30.0</FREQ>
   </TERM>
</ITERATION>
<DOCUMENT>
   <TITLE>Live tables on &amp;lt;b&amp;gt;housing market&amp;lt;/b&amp;gt; and &amp;lt;b&amp;
   </TITLE>
```

```
      <SNIPPET>Table 584: &amp;lt;b&amp;gt;Property&amp;lt;/b&amp;gt; sales based on Land Regist
      </SNIPPET>
      <MODIFIED>1201029162000</MODIFIED>
      <BOOKMARKED>false</BOOKMARKED>
      <URL> http://www.communities.gov.uk/housing/housingresearch/housingstatistics/housingstati
      </URL>
      <SUMMARY>&amp;lt;li&amp;gt;Housing market and house&amp;lt;/li&amp;gt;&amp;lt;li&amp;gt;Ho
      </SUMMARY>
      <ACCESS>1</ACCESS>
      <CREATED>1201030392160</CREATED>
   </DOCUMENT>
   <DOCUMENT>
         <TITLE>Real estate bubble - Wikipedia, the free encyclopedia</TITLE>
         <SNIPPET>A real estate bubble or &amp;lt;b&amp;gt;property&amp;lt;/b&amp;gt; bubble (or
         </SNIPPET>
         <MODIFIED>1200074037000</MODIFIED>
         <BOOKMARKED>false</BOOKMARKED>
         <URL>http://en.wikipedia.org/wiki/Housing_bubble</URL>
         <SUMMARY>&amp;lt;li&amp;gt;Real estate bubbles are invariably followed by severe price
         </SUMMARY>
         <ACCESS>1</ACCESS>
         <CREATED>1201030392161</CREATED>
      </DOCUMENT>
</INTEREST>
</PROFILE>
```

C.2 Profile XML Definition (XSD)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:element name="PROFILE">
   <xs:complexType>
      <xs:sequence>
         <xs:element maxOccurs="unbounded" ref="INTEREST"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="INTEREST">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="NAME"/>
         <xs:element ref="LAST"/>
         <xs:element ref="ACCESSED"/>
         <xs:element ref="ID"/>
         <xs:element maxOccurs="unbounded" ref="ITERATION"/>
         <xs:element maxOccurs="unbounded" ref="DOCUMENT"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="DOCUMENT">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="TITLE"/>
         <xs:element ref="SNIPPET"/>
```

```
            <xs:element ref="MODIFIED"/>
            <xs:element ref="BOOKMARKED"/>
            <xs:element ref="URL"/>
            <xs:element ref="SUMMARY"/>
            <xs:element ref="ACCESS"/>
            <xs:element ref="CREATED"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="NAME" type="xs:string"/>
   <xs:element name="LAST" type="xs:integer"/>
   <xs:element name="ID" type="xs:integer"/>
   <xs:element name="TITLE" type="xs:string"/>
   <xs:element name="SNIPPET" type="xs:string"/>
   <xs:element name="BOOKMARKED" type="xs:boolean"/>
   <xs:element name="MODIFIED" type="xs:integer"/>
   <xs:element name="URL" type="xs:anyURI"/>
   <xs:element name="SUMMARY" type="xs:string"/>
   <xs:element name="ACCESS" type="xs:integer"/>
   <xs:element name="CREATED" type="xs:integer"/>
   <xs:element name="ACCESSED" type="xs:integer"/>
   <xs:element name="ITERATION">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="TIME" type="xs:long"/>
            <xs:element maxOccurs="unbounded" ref="TERM"/>
         </xs:sequence>
      </xs:complexType>
```

```
</xs:element>
<xs:element name="TERM">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="WORD"/>
         <xs:element ref="FREQ"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="WORD" type="xs:string"/>
<xs:element name="FREQ" type="xs:double"/>
</xs:schema>
```