UNIVERSITY OF SOUTHAMPTON

Capturing knowledge of user preferences with recommender systems

by

Stuart Edward Middleton

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

May 2003

UNIVERSITY OF SOUTHMAPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Capturing knowledge of user preferences with recommender systems

by Stuart Edward Middleton

Capturing user preferences is a problematic task. Simply asking the users what they want is too intrusive and prone to error, yet monitoring behaviour unobtrusively and finding meaningful patterns is both difficult and computationally time consuming. Capturing accurate user preferences is, however, an essential task if the information systems of tomorrow are to respond dynamically to the changing needs of their users.

This thesis tests the hypothesis that using an ontology to represent user profiles offers advantages over traditional profile representations in the context of recommender systems.

A novel ontology-based approach to recommendation is applied to a real world problem and empirically evaluated. Synergy between recommender systems and ontologies is then explored to help overcome both the recommender system cold-start problem and the ontology interest-acquisition problem. Finally, the visualization of profiles in ontological terms is examined in a real world situation and empirically evaluated.

List of Contents

List of Figures

List of Tables

Acknowledgements

I would firstly like to thank my supervisor Dave De Roure for his wise advice and foresight in letting me run with my own ideas and allowing them to flourish. I must also thank Nigel Shadbolt for his support of my work and thoughtful ideas. From both Dave and Nigel I have learnt much about how to write papers and perform academic research.

Also of note is Harith Alani, whom I must thank for working with me to integrate his OntoCoPI system with my work. This resulted in a good quality publication and led to a whole new chapter in this thesis.

Lastly, I would like to thank Mark Weal, Chris Bailey, Mark Thompson and Srinandan Dasmahapatra for their help in reading and commenting on the numerous drafts of this thesis. Their selflessness in providing their own time is much appreciated.

Chapter 1 Introduction

Chapter summary
The motivation behind this thesis is described.
The objectives and scope of this thesis are specified.
The structure to this thesis is explained.
The thesis contribution is detailed.
A declaration of originality is made.

1.1 Motivation

Capturing user preferences is a problematic task. Simply asking the users what they want is too intrusive and prone to error, yet monitoring behaviour unobtrusively and then finding meaningful patterns is difficult and computationally time consuming. Capturing accurate user preferences is however, an essential task if the information systems of tomorrow are to respond dynamically to the changing needs of their users.

This thesis examines issues associated with building user profiles for a recommender system by unobtrusively monitoring user behaviour. The thesis explores the idea of using an ontology to represent dynamic user behaviour profiles, and empirically evaluates some of the benefits associated with such an approach. The evaluations presented are grounded in the problem domain of recommending academic research papers.

If the work presented here sheds some light on the path to truly personalized systems it will have been worthwhile.

1.2 Scope and the central hypothesis

1.2.1 Scope

Recommender systems learn their users' preferences over time via both unobtrusive monitoring and relevance feedback. The construction of accurate user profiles is a difficult process. Typically, a recommender system has to work with noisy, indirect evidence of user interest, such as web browsing logs or product purchase records. To

be successful, an appropriate profile representation must be selected for the specific domain, and appropriate profiling techniques employed.

Recommendations are formulated either by finding similar users and recommending the things they liked, collaborative filtering, or by finding similar things to the things a user liked before, content-based filtering. Content-based recommendation normally requires machine-learning techniques to find patterns in the things users like. Collaborative filtering requires a statistical technique to match other user profiles to the profile of the things a user has liked.

As such, the scope of this thesis includes both profiling techniques and machine-learning, in addition to recommender systems.

1.2.2 Central hypothesis

The **central hypothesis** of this thesis is that representing user profiles using an ontology offers advantages over traditional profile representations. Three sub-hypotheses are examined in order to test the central hypothesis.

The **first sub-hypothesis** is that inferences gained about profiles outweigh the loss of information that occurs when a profile is constrained to use only ontological terms. Specifically, the profile accuracy lost by representing profiles in terms of ontological concepts, when compared to a traditional representation, is compensated for by the profile accuracy gained by using the ontology to infer information about a profile.

The **second sub-hypothesis** is that using an ontology allows other knowledge-bases, which use the same ontological concepts, to be used to draw on additional information not normally available to the recommender system. Specifically, using a publications and personnel knowledge-base to bootstrap interest profiles helps reduce the recommender cold-start problem.

The **third sub-hypothesis** is that explicit feedback on user profiles offers advantages over traditional relevance feedback. Specifically, visualizing a profile in ontological terms allows users to explicitly comment on their own profile, providing a rich source of information that will improve profiling accuracy.

If the first sub-hypothesis is true, then an ontological approach to user profiling is at least as good as a more traditional approach. If the second and third sub-hypotheses are true, then an ontological approach to user profiling offers clear advantages, and hence the central hypothesis is true. The objective of the three experiments within this thesis is thus to prove the three sub-hypotheses, and hence the central hypothesis.

1.3 Thesis structure

This thesis begins by introducing recommender systems and the techniques associated with them, setting the context in which this work is set. Three experiments are then discussed, each with the aim of providing supporting evidence to one of the three sub-hypotheses. The evidence is then collated and summarized in order to show support for the sub- hypotheses, and in turn prove the central hypothesis of this thesis.

Chapter 2, Recommender systems, reviews recommender systems in detail by examining the current recommender systems work within the literature. A categorization is proposed for recommender systems, and the reviewed systems placed within this structure.

Chapter 3, Profiling techniques and machine-learning, introduces techniques and concepts in both profiling and machine-learning research that are used by recommender systems.

Chapter 4, The Quickstep recommender system, introduces and describes the Quickstep recommender system. The representations and algorithms are specified, design discussed and experimental details revealed. An ontology is used to represent user profiles, and an evaluation is performed to measure the effectiveness of this approach. Overall usefulness is measured and the system is compared to others in the literature. The conclusions drawn provide evidence to support the first sub-hypothesis.

Chapter 5, Cold-start recommendation and ontology interest acquisition, explores the idea of using both an ontology and a communities of practice identifier to bootstrap the Quickstep recommender system, hence overcoming the classic recommender system cold-start problem. This integration also allows Quickstep to provide up-to-date interests to the ontology, hence addressing the interest acquisition problem

ontologies often have. An experimental analysis is performed and conclusions drawn to provide evidence for the second sub-hypothesis.

Chapter 6, The Foxtrot recommender system, introduces and describes the Foxtrot recommender system. The representations and algorithms are specified, design discussed and experimental details revealed. The idea of visualizing user profiles using ontological terms is explored, allowing users to both see and update their own profiles. An empirical analysis is performed and conclusions drawn to provide evidence for the third sub-hypothesis.

Chapter 7, Conclusions and future work, concludes this thesis by collating the evidence for each sub-hypothesis, and examining how the three sub-hypotheses prove the central hypothesis. The future direction for the work detailed within this thesis is then discussed.

1.4 Contribution

This thesis provides two contributions.

Novel approach

Both the Quickstep and Foxtrot systems use a novel, ontological approach to user profiling within recommender systems. It is novel since most other recommender systems use a binary class approach, using “interesting” and “not interesting” classes specific to each user.

The ontological profile representation used by both Foxtrot and Quickstep represents each profile interest type as an ontological class. The relationships within the ontology are used to infer new knowledge about interests that could not be observed directly. Profiles are also visualized, using the ontological terms that are understandable to users, and profile feedback elicited.

Lastly, the recommender cold-start problem is addressed by applying a knowledge-base, based on publications and personnel data, which uses the same ontology to bootstrap new user profiles.

New evaluation results for an existing concept

The current literature seriously lacks quantitative evaluations of recommender systems addressing real world problems. Both the Quickstep and Foxtrot systems are evaluated using real people while performing a real world task. This in itself is an important contribution to the recommender systems community.

1.5 Declaration

This thesis is based upon work undertaken by the author. What is presented is the original work of the author with the exception of OntoCoPI and the AKT ontology detailed in chapter 5; OntoCoPI and the AKT ontology are part of the advanced knowledge technologies (AKT) EPSRC project at the University of Southampton. This work has been supported by EPSRC studentship grant number 99308831.

Chapter 2 Recommender systems and Ontologies

Chapter summary
The problem domain recommender systems seek to solve is presented.
Common approaches to recommender systems are described, with respect to both the recommendation process and user profiling techniques.
A set of characteristics for recommender systems is defined in terms of the technology and work domain.
A review of the state of the art is conducted, both for commercial systems and those published within the research literature. Systems are categorized according to the previously defined characteristics.
Seminal works are identified and trends in the field discussed.

Recommender systems have become popular since the mid 1990's, offering solutions to the problem of information overload on the World Wide Web. There are several approaches employed, each with its own benefits and drawbacks. Since recommender systems are normally grounded to solve real world problems, the field is both exciting and rewarding to business and academics alike.

In this chapter, current recommender system work is discussed and the approaches used by today's recommender systems described. A set of characteristics for recommender systems is defined and a review of the current state of the art conducted.

Interface agents share many properties with recommender systems, especially in the way they learn about users; interface agents are comprehensively reviewed in [middleton01a].

2.1 The problem of information overload

The mass of content available on the World Wide Web raises important questions over its effective use. With largely unstructured pages authored by a massive range of people on a diverse range of topics, simple browsing has given way to filtering as the practical way to manage web-based information, and this normally means search engines.

Search engines are effective at filtering pages to match explicit queries. Unfortunately, people find articulating what they want as a search query difficult, especially if forced to use a limited vocabulary such as keywords. The result is large lists of search results that contain a handful of useful pages, defeating the purpose of filtering in the first place.

The semantic web offers the potential for help, allowing more intelligent search queries based on web pages marked up with semantic metadata. Semantic web technology is very dependent, however, on the degree to which web pages are annotated by their authors. Annotation requires a degree of selflessness in authors, since the annotations provided will only help others searching their pages. Because of this, and the huge numbers of web pages that require annotation, in the foreseeable future it s likely tat most web pages will remain unannotated. The semantic web will thus only be of partial benefit to the problem of formulating explicit search queries.

2.2 Recommender systems can help

People find articulating what they want hard, but they are very good at recognizing it when they see it. This insight has led to the utilization of relevance feedback, where people rate web pages as interesting or not interesting and the system tries to find pages that match the "interesting", positive examples and do not match the "not interesting", negative examples. With sufficient positive and negative examples, modern machine-learning techniques can classify new pages with impressive accuracy; in some cases text classification accuracy exceeding human capability has been demonstrated [larkey98].

Obtaining sufficient examples is difficult, however, especially when trying to obtain negative examples. The problem with asking people for examples is that the cost, in terms of time and effort, of providing the examples generally outweighs the reward people will eventually receive. Negative examples are particularly unrewarding, since there could be many irrelevant items to any typical query.

Unobtrusive monitoring provides positive examples of what the user is looking for, without interfering with the user's normal work activity. Heuristics can also be applied to infer negative examples from observed behaviour, although generally with less confidence. This idea has led to content-based recommender systems, which

unobtrusively watch user behaviour and recommend new items that correlate with a user's profile.

Another way to recommend pages is based on the ratings provided by other people who have seen the page before. Collaborative recommender systems do this by asking people to rate items explicitly, which allows the system to recommend new items that similar users have rated highly. An issue with collaborative filtering is that there is no direct reward for providing examples since they only help other people. This leads to initial difficulties in obtaining a sufficient number of ratings for the system to be useful, a problem known as the cold-start problem [maltz95].

Hybrid systems, attempting to combine the advantages of content-based and collaborative recommender systems, have also proved popular to date. The feedback required for content-based recommendation is shared, allowing collaborative recommendation as well.

2.3 User profiling in recommender systems

User profiling is typically either knowledge-based or behaviour-based. Knowledge-based approaches engineer static models of users and dynamically match users to the closest model. Questionnaires and interviews are often employed to obtain this user knowledge. Behaviour-based approaches use the user's behaviour as a model, commonly using machine-learning techniques to discover useful patterns in the behaviour. Some sort of behavioural logging is employed to obtain the data necessary from which to extract patterns in behaviour. Kobsa [kobsa93] provides a good survey of user modelling techniques.

The user profiling approach used by recommender systems is behaviour-based, commonly using a binary, two-class model to represent what users find interesting and uninteresting. Machine-learning techniques are then used to find potential items of interest with respect to the binary model. There are a lot of effective machine-learning algorithms based on two classes. Sebastiani [sebastiani02] provides a good survey of current machine-learning techniques, as does chapter 3 of this thesis.

2.4 Features of a recommender system

There are five main issues a recommender system must address; Figure 2.1 lists them all. Firstly, a knowledge acquisition technique must be employed to gather information about the user from which a profile can be constructed. This knowledge is processed to provide the basis for an individual's user profile; it must thus be represented in a convenient way. There must be a knowledge source from which items can be recommended. Recommender systems allow information to be shared amongst users to enhance the overall recommendation performance; this shared information must be clearly defined. The final requirement is for an appropriate recommendation technique to be employed, allowing recommendations to be formulated for each of the users of the system.

Knowledge can either be implicitly or explicitly acquired from the user. Implicit knowledge acquisition is often the preferred mechanism since it has little or no impact on the user's normal work activity. Unobtrusive monitoring of the user discovers behavioural data about the user's normal work activity over a period of time; this data can be used to infer preferences for frequently occurring items. Heuristics can also be employed to infer facts from existing data. Implicitly acquired knowledge requires some degree of interpretation to understand the user's real goals; this is an inherently error prone process, reducing overall confidence in any resulting user profiles.

Explicit knowledge acquisition requires the user to interrupt their normal work to provide feedback or conduct some sort of programming of the system. Explicit knowledge is generally high confidence information, since it is provided by the users themselves and not acquired from indirect inference. Feedback types include item relevance, interest and quality. User programming occurs when the user is asked to create filter rules, either visually or via a programming language, or to tell the system about groups or categories of items that exist in the domain.

User feedback can be shared for the purpose of recommendation. If collaborative filtering is to be used, other users' feedback on unseen items can be used as a basis for recommendations for a particular user. Examples of interesting items can be shared between similar users to increase the size of the training set and hence improve classification accuracy. Previous navigation patterns are also useful to share, as they

allow new users to receive the benefit from other people's previous mistakes and successes.

Domain knowledge can also be shared, since it is normally programmed in and hence available to the system from the start. Categorizations of items can be used to provide order to a domain, and common sets of domain heuristics, potentially part of a knowledge base, can be useful when computing recommendations.

Profiles can be represented as a feature vector in a vector-space model. This is a standard representation and allows easy application of machine-learning techniques when formulating recommendations. For content-based recommendation the features in the vectors might be the word frequencies of interesting documents, while for collaborative filtering the features could be the keywords commonly used by users in their search queries. Navigation trails can be used to represent time-variant user behaviour. If some initial knowledge engineering has been conducted there may also be knowledge about the users available to a profile.

The domain itself will contain sources of information to be recommended to the users. These could be from a database held by the recommender system, such as movie titles, or available dynamically via the web, such as links from the currently browsed page or web pages crawled from a web site. Systems can also rely on external events, such as incoming emails, to provide items for recommendation.

There is a wide variety of recommendation techniques employed today, with most techniques falling into three broad categories. Rule filters apply heuristics to rank items in order of potential interest. Machine-learning techniques employ similarity matching to rank items in order of interest. Collaborative filtering finds similar users and recommends items they have seen and liked before.

To summarize, the set of features important to a recommender system are:

- Knowledge acquisition technique
  - Implicit
    - Monitoring behaviour
    - Heuristics to infer information
  - Explicit
    - User feedback
    - User programming
      - Filter rules
      - User-created groups/categories
- Shared information
  - User feedback
    - Item feedback
    - Examples of items
    - Navigation history
  - Domain knowledge
    - Item groups / categorizations
    - Domain heuristics
- Profile representation
  - Vector model
  - Navigation trails
  - Knowledge-based profile
- Knowledge source
  - Internal database of items
  - Crawled web pages
  - External domain events
- Recommendation technique
  - Heuristics
  - Similarity matching
  - Collaborative filtering

Figure 2.1 : Technology features of a recommender system

One line definitions of each technology feature are available in table 2.1.

Technology feature	One-line description
Monitoring behaviour	System observes users using it and records this behaviour.
Heuristics to infer information	Rules are used to infer information about users
User feedback	Users provide explicit feedback e.g. item relevance, item examples, etc.
Filter rules	Users provide filter rules to the system
User-created groups/categories	Users define system groups or categories
Item feedback	Item feedback is used by the system to help other users
Examples of items	System pools examples of items to form a collective training set
Navigation history	System uses recorded navigation histories to help other users
Item groups / categorizations	System shares communal groups and categories, whether defined by the system or other users
Domain heuristics	System shares a set of domain filter rules between all users
Vector model	System uses vectors to model for documents or interest profiles
Navigation trails	System holds a navigation history e.g. a web browsing history
Knowledge-based profile	System uses knowledge-based profiles
Internal database of items	System recommends from an internal database of items
Crawled web pages	System crawls the web for items to recommend
External domain events	Events occur that trigger recommendation e.g. an email arrives
Heuristics	Rules are used to find best items to recommend
Similarity matching	Similarity function is used to find items matching a content-based profile
Collaborative filtering	Statistical functions are used to find people with similar profiles, then items liked by those people are recommended

Table 2.1 : Technology feature one-line definitions

2.5 Classification by technology

Table 2.2 lists the recommender systems reviewed in appendix A and shows how they are classified according to the technological features identified previously; forty-four recommender systems are reviewed in total. Entries in this table are sorted by knowledge-acquisition technique and entries marked by a "-" are due to the information being unavailable. It is common for commercial systems to withhold their technologies to maintain commercial advantage. This table provides a clear representation of the recommender system domain as it is today. Commercial systems are highlighted in a bold typeface.

The Quickstep and Foxtrot recommender systems are included within table 2.2 to show where they fit in the literature. Details of these experimental systems can be found in chapters 4 to 6.

	Knowledge acquisition technique					Shared information					Profile representation			Knowledge source			Recommendation technique		
	Monitoring behaviour	Heuristics to infer information	User feedback	Filter rules	User-created groups/categories	Item feedback	Examples of items	Navigation history	Item groups/categories	Domain heuristics	Vector model	Navigation trails	Knowledge-based profile	Internal database of items	Crawled web pages	External domain events	Heuristics	Similarity matching	Collaborative filtering
CoCoA	o		o			o			o	o			o	o				o	o
Comm'ty search ass'nt	o							o				o		o				o	
ELFI	o						o				o			o				o	
FAIRWIS	o		o			o					o		o	o					o
Foxtrot	o	o	o			o	o	o	o		o				o			o	o
Ghani	o						o		o		o				o			o	
MEMOIR	o		o					o				o			o				o
OWL	o					o								o					o
ProfBuilder	o							o			o				o			o	o
QuIC	o						o		o		o			o	o			o	
Quickstep	o	o	o		o	o	o	o	o		o				o			o	o
Referral Web	o	o							o						o	o	o		
SOAP	o		o			o					o				o				o
SurfLen	o							o				o			o			o	o
Tapestry	o		o	o		o		o		o						o	o		
Entree		o					o		o	o			o	o				o	
PHOAKS		o							o	o				o			o		
Amazon.com			o			o		o	o		-	-	-	o					o
Campiello			o			o			o		o			o					o
CBCF			o				o		o		o			o				o	o
CDNOW			o			o		o	o		-	-	-	o					o
Dietorecs			o						o	o			o	o				o	
eBay			o			o			o		-	-	-	o					o
EFOL			o		o	o			o					o					o
Expertise Recom'er			o	o	o	o				o	o		o	o			o		
Fab			o			o			o	o	o				o			o	o
GroupLens			o			o			o		o			o					o
ifWeb			o			o					o				o			o	
Levis			o			o			o		-	-	-	o					o
LIBRA			o				o		o		o			o				o	
METIOREW			o				o				o			o					o
MIAU			o						o				o	o					o
Moviefinder.com			o			o			o		-	-	-	o					o
MovieLens			o			o			o		o			o					o
Nakif			o			o					o			o					o
P-Tango			o			o			o		o			o				o	o
RAAP			o			o			o		o			o				o	o
Recommend' Explorer			o				o		o		o			o				o	
Reel.com			o			o			o		-	-	-	o					o
RIND			o				o		o		o			o				o	o
Ringo			o			o			o		o			o					o
Siteseer			o			o					o			o					o
Ski-europe.com			o			o					o			o				o	o
Virtual rev's (Tatemura)			o		o	o			o		o			o					o

Table 2.2 : Classification of recommender systems by technology

From this classification we can see that most recommender systems today explicitly ask users for feedback and share that feedback between users to provide collaborative recommendation from an internal database of items.

This trend is understandable when you consider the technology available. Eliciting user feedback is an optional task and normally requires minimal effort from the user if

eliciting at the point of recommendation. The feedback device is often associated with a reward to encourage participation, for example a feedback control next to a recommended web page.

Casual users are often reluctant to either register or install software, making monitoring of their behaviour difficult. It is also unlikely that users will accept the invasion of privacy that occurs with monitoring unless the reward offered is substantial. Heuristics can successfully infer information about users but they normally need data to work on, be it from existing logs or monitored behaviour. Filter rules are generally too complex to define in any useful detail, and creating categories or groups for a system requires a substantial investment of human effort for little immediate reward.

Perhaps the most compelling evidence for the benefits of a simple feedback/collaborative filtering approach is the marketplace. All the commercial recommender systems reviewed use this technology. It thus appears clear that this approach is the only one to have yet reached maturity. This is not the full story, however. While unsuitable for the type of mass-market operation the commercial systems are targeting, other approaches would work for a smaller user base from corporations down to a groups of individuals.

Monitoring user behaviour is useful in a corporate environment, where software is installed for everyone and computers are used for work purposes only. Several of the reviewed systems do indeed use monitoring techniques, which tend to share navigation histories and implicit user feedback. Here, a mix of collaborative and content-based approaches to recommendations are seen, with the content-based systems using vector-space models to perform similarity matches between the domain content and user profiles.

For information systems with large databases, heuristics can be used to infer knowledge about the users. If a large corpus of information exists of users and their relationships, this could be mined and recommendations generated from it. User relationships can be found from email messages, newsgroup archives mined for web references, bookmarks utilized etc. These systems tend use filter rules to select

appropriate items for recommendation, avoiding the need for user feedback completely.

2.6 Classification by work domain

Figure 2.2 lists the recommender systems reviewed in appendix A by work domain. This classification provides an overview of the different types of work domain current recommender systems address. Commercial systems are highlighted in a bold typeface.

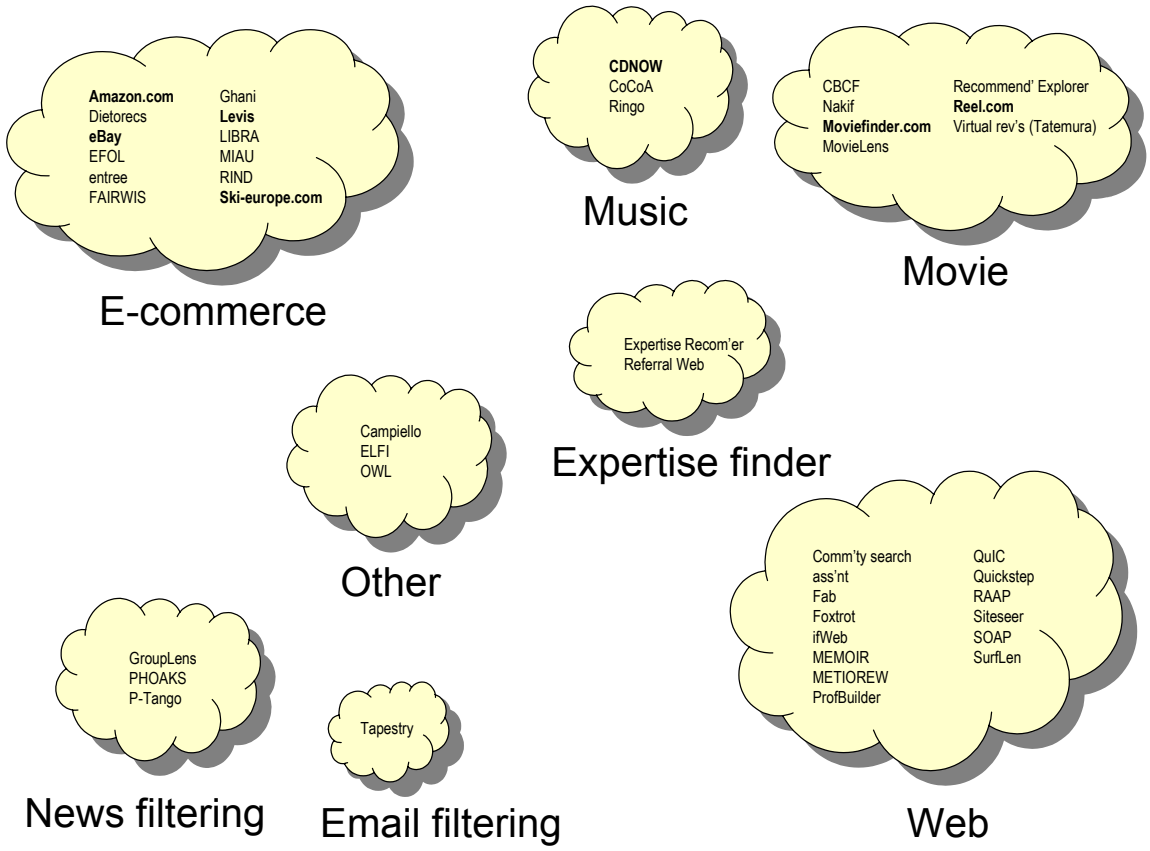


Figure 2.2 : Classification of recommender systems by work domain

The majority of systems lie in either the web domain or the e-commerce domain. For commercial systems only the e-commerce, music and movie domains are used. This reflects the reality that successful commercial recommender systems must offer the customer a service they value enough to invest time and money. E-commerce and music systems tend to sell products while movie systems tend to make money from web advertising to repeat users. Perhaps the failure of web recommender systems to make a significant commercial impact reflects the immaturity of the technology

behind them, and the difficulty of the technical problems they face to make them good enough in the marketplace.

2.7 Seminal recommender systems

The first recommender system, Tapestry [goldberg92], coined the phrase "collaborative filtering" that has been used by many others since. Seminal examples of collaborative recommender systems include GroupLens [konstan97] and PHOAKS [terveen97], both published in the influential 1997 special issue of the Communications of the ACM on recommender systems. Content-based recommender systems are exemplified by Fab [balabanović97], a seminal hybrid recommender system.

Other systems of note are SOAP [voss97], which was one of the first recommender systems to use an agent metaphor and ReferralWeb [kautz97], which looked into recommending people with useful expertise.

2.8 Recommender system examples

The GroupLens [konstan97] recommender system is a classic recommender system which recommends Usenet newsgroup articles. As users browse their usenet news a split screen interface presents some recommendations, as shown in figure 2.3, along with the ratings provided by other GroupLens users. The aim of the split screen interface is to blend into the normal usenet interface. With 50,000+ new messages posted each day this recommendation interface provides some way to identify what is worth reading and what is not.

CDNOW is a commercial music CD shop/recommender system. It is now integrated into the wider application of Amazon.com. Customers buy CD's through a standard electronic shop web-based interface as shown in figure 2.4. As customers shop, by navigating through the sites web pages, CDNOW presents opportunistic recommendations of items that the user might want. These recommendations are based on the previous navigation pattern and buying habits of the customer.

If a customer wants they can provide feedback as to which artists they prefer and own. Likes and dislikes can be indicated and a set of 6 albums recommended upon request. Feedback on these recommendations is also elicited.

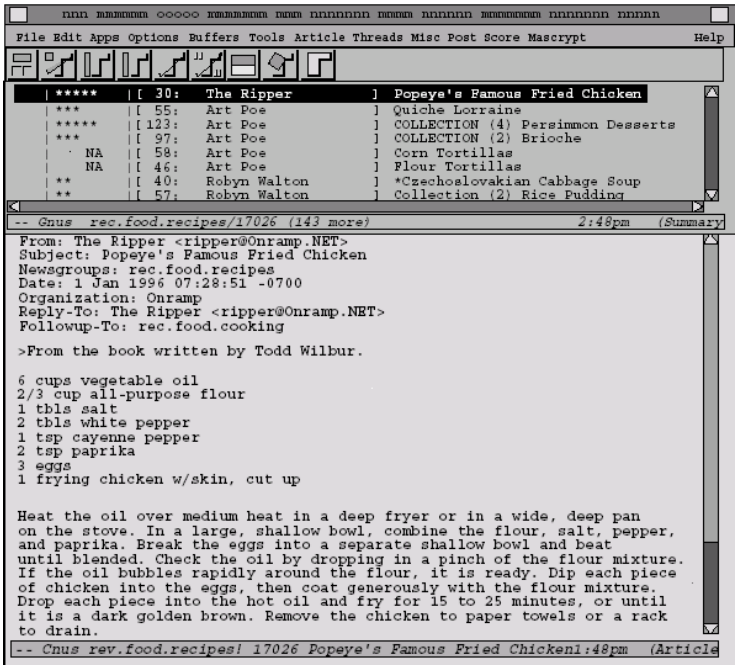


Figure 2.3 : GroupLens interface

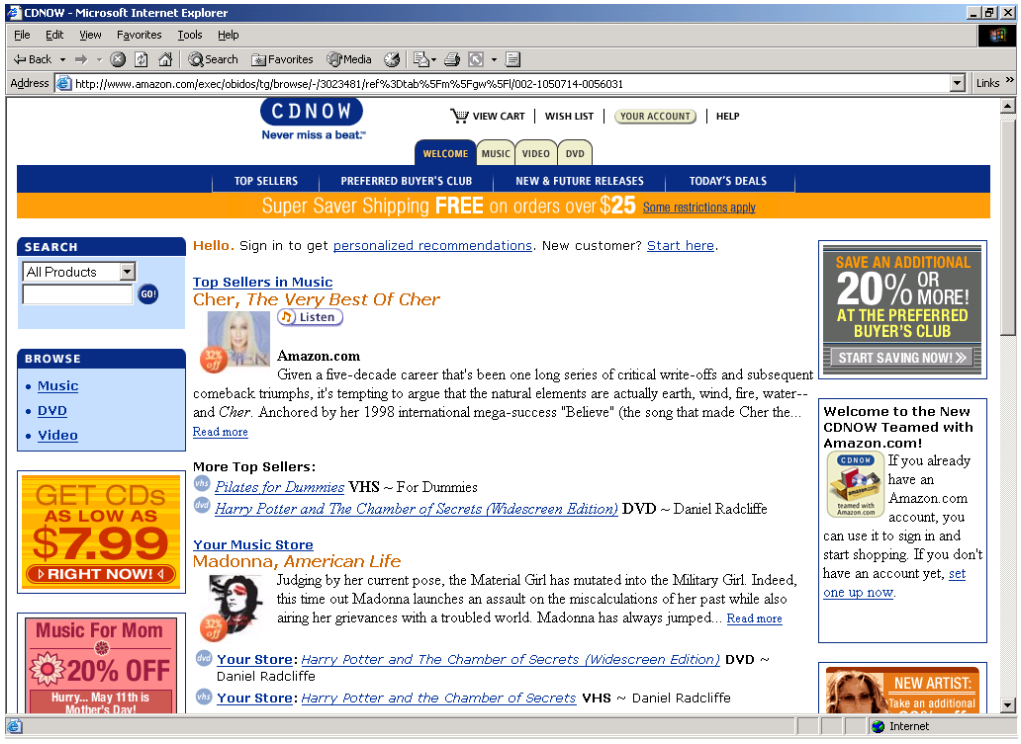


Figure 2.4 : CDNOW interface

2.9 Ontologies

An ontology is a conceptualisation of a domain into a human-understandable, but machine-readable format consisting of entities, attributes, relationships, and axioms *[guarino95]*. Ontologies can provide a rich conceptualisation of the working domain

of an organisation, representing the main concepts and relationships of the work activities. These relationships could represent isolated information such as an employee's home phone number, or they could represent an activity such as authoring a document, or attending a conference.

Ontologies are normally used within knowledge-based systems to define the classes and structure of the domain models supported. The experimental recommender systems in this thesis use ontologies in such a way, defining the classes used to represent research paper topics and the structure of the generalization relationships between these classes. Ontologies are employed in such a way that we can further employ inference, mappings to external knowledge-bases and mappings to human understandable concepts.

The two experimental recommender systems described in this thesis represent user profiles using the classes defined in the system's research paper topic ontology. The user profiles thus form a dynamic knowledge-base, holding interest information about the users of the recommender system. This profile knowledge-base is used to drive the recommendation process. Further to just recommendation, however, the ontological representation of user profiles allows inference of interests via the is-a relationships defined within the ontology, bootstrapping from information gleaned by mappings to an external ontology and profile visualization based on the ontological profile classes.

In the three experiments described in this thesis only a simple is-a taxonomy is used; this is perhaps one of the simplest ontological representations. The reason for this simplicity is to reduce the number of potential causes of experimental observations to a minimum. However, many more relationships and concepts could be employed in the recommender system ontology than just research paper topics and generalization relationships. The second experiment in this thesis alludes to some of the potential benefits of a more capable ontology, making use of project and personnel knowledge held in external sources. As discussed in the concluding chapter, expanding the ontology by including additional concepts and relationships is a clear way to increase the potential for inference and hence improve profiling accuracy.

2.10 Ontology example

An example of an ontological system is the ScholOnto system *[shum00]*. ScholOnto is a digital library of academic research papers which allows researchers to semantically mark-up papers and share this mark-up for collaborative interpretation and discourse. ScholOnto maintains an ontology supporting claims made by researchers about papers; claims in this context are personal assertions of relationships between supported concepts and/or other claims. The concepts supported by this ontology include a papers "approach", "ideas", "language" and "methodology"; the relationships supported include "uses", "envisages", "confirms" and "raises problem". The ontology is thus well suited to represent scholarly discourse over the significance of ideas and concepts within their field.

ScholOnto's ontology is presented to the users via HTML web pages, generated by a knowledge representation tool. These web pages allow users to see the complex relationships between each ontological concept. A schema language allows rules to be created to operate on the supported knowledge model; such rules can be used to signal inconsistencies within a model.

Various interface views to this knowledge are supported, including a conventional form/menu interface, direct annotation interface and semantic network interface. These interfaces make it possible for researchers to see different perspectives on a body of research, such as a "European" perspective on EU driven research work.

There is a significant need for researchers to provide the initial knowledge capture to populate the knowledge models. This is made an easier task with the helpful interfaces, but the authors will be looking into methods such as machine learning to automatically detect perspectives within the ontology.

2.11 Conclusion

Current recommender systems use a wide variety of techniques, often adopting a hybrid approach with both collaborative and content-based filtering techniques being employed. The commercial systems, however, use a smaller subset of the potential techniques available, concentrating on offering a collaborative filtering service to sell products.

The Quickstep and Foxtrot experimental systems described later in this thesis use a hybrid approach, using both similarity matching and collaborative filtering. They unobtrusively monitor users in addition to eliciting feedback, forming a basis upon which to run advanced profiling algorithms and techniques. The classification in table 2.2 and figure 2.2 clearly places these systems in the context of the recommender system literature.

The next chapter describes in detail the profiling and machine-learning techniques used by recommender systems in the literature.

Chapter 3 Profiling techniques and machine-learning

Chapter summary
The profiling representations used by recommender systems are discussed.
The profiling techniques used with these representations are detailed.
The machine-learning algorithms commonly used in recommender systems are described.

This chapter introduces the machine-learning and profiling techniques commonly employed by recommender systems, along with the profile representations that they use. Many techniques have been published in the user-modelling [kobsa93] and machine-learning [sebastiani02] literature. The techniques described here are the subset commonly used by the recommender systems community. Generally speaking, once a profile representation is decided upon, the technique that you can apply to that representation can be chosen.

3.1 Backus-Naur Format

This thesis uses a non-rigorous version of Backus-Naur format (BNF) in the figures to describe data formats. Figure 3.1 briefly describes BNF for easy reference.

Data types represented by angled brackets	<>
Items are assigned a data type	item = <data type>
Sets of items are enclosed by parentheses	set = $(item_1,item_2 \ldots item_n)$
List of 0 or more items use a *	list = $(item_1)^*$
	= $(item_1)(item_1)\ldots(item_1)$
List of 1 or more items use a o	list = $(item_1,item_2)^o$
	= $(item_1,item_2)(item_1,item_2)\ldots(item_1,item_2)$

Figure 3.1 : Backus-Naur format

3.2 Profile representations

Profile representations falls into two types, which are not mutually exclusive. Ratings-based representations store every user's ratings on available items so correlation techniques can be used to find similar users. Content-based representations store representations of specific items of interest to a single user so machine-learning techniques can find similar items.

3.2.1 Ratings-based representations

Relevance feedback

When users receive recommendations it is common to elicit feedback on how interesting the recommendations are to the needs of the user. This type of feedback is called relevance feedback.

Relevance feedback is elicited by offering the user a rating scale for each recommendation; the choice is commonly either "interesting" and "not interesting" or a 3 to 5-point scale of interest. The representation of relevance feedback is thus a set of recommended items and the associated interest values provided by each user. Relevance feedback is often incomplete since users are often reluctant to invest time and effort to provide the feedback.

Relevance feedback can be acquired implicitly, allowing inference from observed user behaviour. The problem with implicit feedback is that the assumptions made to allow inference often introduce errors. For example, a user may read an initially interesting looking document, only to find it was actually not interesting after all when its details are known; if all documents that are read are inferred to be interesting this situation would clearly introduce an error into the relevance feedback acquired. Implicit feedback is commonly in a positive/negative form, implied by clear positive or negative actions in an attempt to reduce the number of assumptions required.

A balance must be made between interrupting the user to acquire high quality explicit feedback and unobtrusive methods to obtain lower quality implicit feedback. Exactly how much interruption users will tolerate will depend upon the specific application domain.

3.2.2 Content-based representations

Most content-based analysis is performed on textual documents such as web pages, newspaper articles or document abstracts. The reason for this is that textual documents easily break down into individual words, whereas video and audio sources require sophisticated analysis to decompose into useful sub-components. All content-based recommender systems work with textual content.

Term-frequency vector representation

The most common abstraction of a textual document in the machine-learning context is a term-frequency (TF) vector. Terms consist of single words or phrases, and the frequency count is simply the number of times a term appears within the document text. To create a term-frequency vector the terms within a document are counted and the frequency values stored in an n-dimensional vector. The number of dimensions of the vector is the number of unique terms within a document.

It is common to reduce the dimensionality of term-frequency vectors to improve processing efficiency. Common terms, called stop words, are removed since they have little discriminating power as all documents contain them; examples of stop words are "and", "if" and "the". The removal of stop words is normally performed using a standard stop list, removing all terms that match the stop list.

Low frequency terms are also removed, since they too have little discriminating power, often appearing in just one document; an example of low frequency term is a web URL.

Another dimensionality reduction technique commonly employed is to stem terms. This involves removing suffixes so that basically similar words are grouped together; an example would be to use the stemmed term "recommend" for the terms like "recommender", "recommendation" and "recommends".

In practice, stemming, stop lists and low frequency term removal are all applied to reduce the dimensionality of the term vectors as much as possible. Sebastiani [sebastiani02] covers the subject of dimensionality reduction in more detail.

Term-frequency representations are often called "bag of words" representations, since the structure of the document is lost. It has been shown [lewis92] that the loss of structural information such as sentences and paragraphs does not significantly degrade the performance of subsequent analysis and classification.

Recommender systems usually normalize the frequency data based on the length of the document, and some systems weight individual terms in favour of the more discriminating ones. This avoids larger documents always having highly weighted terms.

Binary class profile representation

The most common profile representation for content-based recommender systems is the binary class profile, representing user interests as a set of positive and negative examples. The positive, or "interesting", examples are represented as a collection of term-frequency vectors of documents that the user has rated as "interesting". The negative, or "not interesting", examples are likewise represented. This binary class representation is very suitable for a great many machine-learning techniques.

Since relevance feedback is required to obtain the sets of positive and negative examples, a ratings-based profile is often additionally implemented to create a hybrid recommender system.

Multi-class profile representation using an ontology

The alternative to the binary class representation is a multi-class representation. Rather than simply having positive and negative classes, an ontology of classes can be created that map to domain concepts such as newspaper topics like "sport". A user's profile is thus represented in terms of which classes they are most interested in, abstracting away from the specific examples of interest. When relevance feedback is acquired, examples of interest are classified according to the classes within the ontology, and the user's interest in that class recorded.

Multi-class classification is considerably more complex than binary class classification. Having more than two classes reduces the number of examples available for each class, thus reducing the accuracy of the machine-learning technique employed. In addition, since classes are shared between users, there will be a loss of information about individual user interests when compared to a binary representation where each user has their own set of examples; sharing examples does allow for a larger training set, however. These factors are the reason why very few recommender systems adopt this approach.

Most ontologies are created manually by a knowledge engineer and domain experts. They thus capture the relevant classes within a domain and relationships between them. It is possible to create classes automatically using clustering machine-learning algorithms. Clustering finds similar term frequency vectors and groups them together

to make a class. Classes created by clustering, however, have no domain knowledge associated with them, making useful inference from them difficult.

3.2.3 Knowledge-based profile representation

Knowledge-based profile representations appear in the user modelling literature. Typically these approaches require questionnaires and interviews with users to acquire information about their requirements before a profile can be built. Profiles consist of asserted facts about a user in a knowledge-base, from which inferences can be drawn about user stereotypes and interests. Knowledge-based profiles are often used in the related fields of agent and intelligent tutoring systems, however. Kobsa [kobsa93] provides a good introduction to the field of user modelling and profile representation.

3.3 Profiling techniques

3.3.1 Time-decay functions

Time decay functions are simple profiling techniques, and can be applied to content and rating-based profile representations so long as their information is time-stamped. A weighting function is defined which contains an inverse time weight, so older information is less relevant than more recent information. This weighting function is then applied to one of the rating values, term weights or class interest values.

Due to their simple nature, time decay functions can be successfully applied to very complex profiles, where multiple patterns exist and evidence is incomplete.

$$w(t_i) = \sum_{j=1}^{N} \frac{tf(t_i,d_j)}{age(d_j)}$$

$w(t_i)$	weight of term t_i after time decay
t_i	i^{th} term
N	number of documents
$tf(t_i,d_j)$	number of times term t_i appears in document d_j
d_j	j^{th} document
$age(d_j)$	age of document d_j

Figure 3.2 : Time decay function

3.3.2 Pearson-r correlation

The Pearson-r correlation algorithm is the most common algorithm used on rating-based representations to find similar people to a given user. Pearson-r correlation finds correlations between different user ratings on particular items. The users with the highest Pearson-r values have ratings most similar to the target user.

$$r_{xy} = \frac{\sum_{i=1}^{N} (U_{xi} - \bar{U}_x) * (U_{yi} - \bar{U}_y)}{\sqrt{\sum_{i=1}^{N} (U_{xi} - \bar{U}_x)^2 * \sum_{i=1}^{N} (U_{yi} - \bar{U}_y)^2}}$$

r_{xy}	pearson-r correlation between user x and y
N	number of ratings
U_{xi}	i^{th} rating for user x
$\bar{U}_x$	mean rating for user x

Figure 3.3 : Pearson-r correlation

$$r_{xy} = \frac{\sum_{i=1}^{N} (U_{xi} - \beta) * (U_{yi} - \beta)}{\sqrt{\sum_{i=1}^{N} (U_{xi} - \beta)^2 * \sum_{i=1}^{N} (U_{yi} - \beta)^2}}$$

r_{xy}	pearson-r correlation between user x and y
N	number of ratings
U_{xi}	i^{th} rating for user x
β	constrained value

Figure 3.4 : Constrained Pearson-r correlation

The Pearson-r correlation algorithm adjusts to bias within ratings, such as from users who always rate highly, by computing the deviation of ratings from the mean rating of each user. If ratings have no bias a constrained Pearson-r algorithm can be used, with the β coefficient set to the threshold level of a "good" rating. Pearson-r correlations suffer from the cold-start problem, where initial results are poor until enough ratings have been accumulated with which to form significant correlations.

3.3.3 Other profiling techniques

Other profiling techniques include time-series profiling algorithms such as piece-wise representation and curve fitting functions. Piece-wise representation [keogh99] splits a time-series profile into slices and matches newly seen data to these previous slices. Predictions for future activity are modelled on the most similar slice. Curve fitting, such as mathematical polynomial methods, models the previously seen activity and uses the model to predict future activity.

Knowledge-based profiling techniques also exist where knowledge is acquired about user preferences and experience, often via interviews or questionnaires, and modelled in a knowledge base. Contextual questions can then be asked about users, and reasoned predictions inferred from known domain knowledge. Stereotyping [rich79] is an example of a knowledge-based profiling technique.

3.4 Machine-learning techniques

Machine-learning (ML) techniques fall into two categories according to whether they require a labelled set of examples or not. Supervised learning takes a set of labelled example cases, called a training set, as the basis for categorization of new cases. Unsupervised learning does not use a training set since classes are generated from patterns within unlabelled examples.

Supervised learning is more accurate since it avoids the errors introduced by automatic class generation. Labelling a training set is, however, a time-consuming task often performed by hand. Due to the nature of relevance feedback, recommender systems use supervised learning techniques.

Most machine-learning algorithms deal with the binary class case and generalize to the multi-class case. Only a few algorithms are designed with the multi-class case in mind.

3.4.1 Data mining

With a large dataset of user behaviour, data mining techniques can be used to discover patterns of behaviour. One such technique is the A-Priori [agrawal94] algorithm, which learns association rules from a large dataset of transaction data such as supermarket shopping data. Each pass of the data counts the support for individual

items and determines which item sets are well supported. These sets are then used as a seed to find new, well supported itemsets called candidate itemsets. This continues until no new itemsets are found.

```
for (k=2;L_{k-1}≠0;k++) {
        C_k = apriori-gen(L_{k-1})
        forall transactions t ∈ D {
                C_t = subset( C_k,t )
                forall candidates c ∈ C_t {
                        increment count for c
                }
        }
        L_k = { c ∈ C_k | c count ≥ min }
}
answer = union of L_k
```

L	well supported itemset = (item,count)*
L_k	k^{th} item in well supported itemset
C	candidate itemset = (item,count)*
C_k	k^{th} item in candidate itemset
D	database of user transactions
subset(c,t)	hash-tree lookup to find candidates in set c for a transaction t
apriori-gen(l)	algorithm to generate candidate sets from itemset l

Figure 3.5 : A-Priori algorithm

3.4.2 Information theoretic methods

Term frequency-inverse document frequency (TF-IDF) is the most popular information theoretic method [van rijsbergen79] for recommender systems.

$$w(t_i,d_j) = tf(t_i,d_j) * \log \frac{N}{df(t_i)}$$

$w(t_i,d_j)$	tf-idf weight of term t_i in document d_j
t_i	i^{th} term
d_j	j^{th} document
$tf(t_i,d_j)$	number of times term t_i appears in document d_j
N	number of documents
$df(t_i)$	number of documents containing term t_i

Figure 3.6 : Features used in TF-IDF algorithm

This term weighting biases the chosen classifier towards terms which have a high frequency count within their documents, but only appear in a small number of documents. These terms are likely to be good discriminators.

Once the term weights are calculated a vector similarity measure is applied to new documents to determine classification. A typical vector similarity measure is the cosine similarity measure, or dot product.

$$sim(d_a,d_b) = \sum_{t \in d_a} w(t,d_a) * w(t,d_b)$$

$$w(t,d) = \frac{tfidf(t,d)}{\sqrt{\sum_{i=1}^{T_d} tfidf(t_i,d)^2}}$$

$sim(d_a,d_b)$	similarity measure between document a and b
d_a,d_b,d	document vectors
t	term
$w(t,d)$	normalized cosine weight
$tfidf(t,d)$	tf-idf weight of term t in document d
T_d	number of terms in document d

Figure 3.7 : Features used in Cosine similarity algorithm

The Rocchio algorithm [van rijsbergen79] is another information theoretic algorithm used by recommender systems. Rocchio takes positive and negative examples and computes the relevance of each term in the form of a class vector.

$$class_i = (w_{1i}, \ldots w_{Ti})$$

$$w_{ki} = \frac{\beta}{N_{positive}} \sum_{j=1}^{N_{positive}} w_{kj} - \frac{\gamma}{N_{negative}} \sum_{j=1}^{N_{negative}} w_{kj}$$

$class_i$	term vector for class i
w_{ki}	weight for term k document i
T	number of terms in document set
β, γ	constants, typically β=4, γ=16
$N_{positive}$	number of positive documents
$N_{negative}$	number of negative documents

Figure 3.8 : Features used in Rocchio algorithm

These class vectors are then used with a cosine similarity measure to classify new documents. Computing the term weights is time consuming, but once done classification is fast. If the training set changes then the weights must be re-computed.

Latent semantic indexing (LSI) [deerwester90] is a term weighting technique that must be combined with a similarity matching technique such as cosine-similarity.

Document vectors are compressed into a low dimensional space based on patterns of word co-occurrence found after term-document matrix manipulation using eigenvectors. Terms below a threshold are removed from the final document vectors, thus reducing the dimensionality of the final document-vector space. It is meant to bring out the "latent" semantic structure of the vocabulary used in a document corpus.

3.4.3 Instance-based methods

Techniques that hold the training set in memory are instance-based methods. The most common instance-based method in the recommender systems literature is the k-nearest neighbour (kNN) algorithm [mitchell97]. The kNN algorithm stores all example vectors within a term-vector space and computes the distance between these example vectors and an unclassified vector. The distance measure is normally the cosine similarity measure, and an inverse distance weighting is often applied to reduce the effect of distant neighbours. Inverse weighting is particularly useful where there is a very close neighbour and many unrelated neighbours; since k neighbours are taken the distant ones will tend to dominate unless an inverse distance weighting is applied. The classification confidence in a class is the sum of the distances from the closest k examples of that class.

$d_1 \ldots d_k$ are the k nearest documents to d_{new}

$$f(d_{new}) = \sum_{i=1}^{k} \frac{1}{w(d_{new}, d_i)}$$

$$w(d_a, d_b) = \sqrt{\sum_{j=1}^{T} (t_{ja} - t_{jb})^2}$$

$w(d_a,d_b)$	kNN distance between document a and b
d_a,d_b	document vectors
T	number of terms in document set
t_{ja}	weight of term j document a
f(d)	k-NN function

Figure 3.9 : Features used in kNN algorithm

Since new instances can be added to the vector space without the need to re-compute weights, the set-up time is minimal. Classification can be time-consuming, however, since all example distances must be computed for every unclassified vector. The memory requirement is proportional to the size of the training set.

Where instances are held in a more symbolic representation, case-based reasoning can be used. Classification involves a search of known instances to find those cases that have the greatest number of matching parts. Knowledge-based reasoning can be used to combine cases to construct a response to a specific query and problem-solving knowledge can be attached to specific cases. An example of case-based reasoning is the CADET [sycara92] system.

3.4.4 Probabilistic methods

Methods that compute the probability of a vector belonging to a particular class, based on the set of terms, are probabilistic methods. A large network of probabilities is normally computed from the training set, showing the likelihood of a class given a set of terms. Bayes theorem is the basis for many probabilistic techniques, with the naïve Bayes classifier [mitchell97] being popular for recommender systems.

$$P(h|D) = \frac{P(D|h)\,P(h)}{P(D)}$$

P(h|D)	probability of h given D
h	hypothesis
D	data
P(h)	probability of h

Figure 3.10 : Bayes theorem

The naïve Bayes classifier uses Bayes theorem and an assumption of conditional independence of terms given the classification label. This assumption is incorrect, but the algorithm has been shown [domingos97] to give good results anyway. Zero term probabilities, resulting from zero term frequencies, also present a problem since probabilities are multiplied together. As such, zero probabilities are often replaced by an estimate based on sample size [mitchell97].

$$P_c(d_j) = \underset{d_j \in D_c}{\text{argmax}}\ P(d_j) \prod_{i=1}^{T} P(t_i|d_j)$$

$P_c(d_j)$	probability of document d_j belonging to class c
d_j	j^{th} document
D_c	document set for class c
$P(d_j)$	probability of document d_j occurring
T	number of terms in document set
$P(t_i|d_j)$	probability of term t_i occurring in document d_j

Figure 3.11 : Features used in Naive Bayes classifier

Where the number of training examples is limited, but there are many unlabelled examples, Expectation-Maximization (EM) [mitchell97] can be used. The EM algorithm searches for the maximum likelihood for an hypothesis by seeking the hypothesis that maximizes the expected value given a data set, consisting of labelled and unlabelled data. The expected value is computed from the probable full dataset, so hypotheses are found that maximize this value.

$$Q(h'|h) = E[\ \ln P(Y|h')|h, X\ ]$$

Step 1 : Estimation

Calculate Q using the current hypothesis h and labelled data X
to estimate the probability distribution over Y
$Q(h'|h) \leftarrow E[\ \ln P(Y|h')|h, X\ ]$

Step 2 : Maximization

Replace hypothesis h by the hypothesis h' that maximizes the Q function
$h \leftarrow \underset{h'}{\text{argmax}}\ Q(h'|h)$

h	current hypothesis
h'	new hypothesis
E	expected value
P(a,b)	probability of a given b
Y	full dataset
X	labelled dataset

Figure 3.12 : EM algorithm

Probabilistic methods require a significant set-up time to build the probabilistic network, and it must be re-formulated if the training set changes. Once built however, the classification time is minimal.

3.4.5 Boosting and bagging

Boosting works by repeatedly running a weak learning algorithm on various distributions of the training set, and then combining the classifiers produced by the weak learner into a single composite classifier. The "weak" learning algorithm is often actually a strong binary algorithm such as a naïve Bayes, kNN and decision trees.

Classification error is evaluated per iteration, and used as the basis for the specialization of that iteration's classifier weights. The final classifier takes the weighted votes of each iteration's classifier, and returns the class with the highest total vote. AdaBoostM1 is an example of a boosting algorithm designed to work with multiple classes.

Initialise all values of D to 1/N
Do for t=1..T
    call weak_learn(D_t)
    calculate error e_t
    calculate β_t
    calculate D_{t+1}

$$e_t = \sum_{i = \text{all incorrect classes}} D_t(i)$$

$$\beta_t = e_t/(1-e_t)$$

$$\text{classifier} = \underset{c \in C}{\text{argmax}} \sum_{t = \text{all iterations with result class c}} \log \frac{1}{\beta_t}$$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if classification correct} \\ 1 & \text{if classification incorrect} \end{cases}$$

D_t	class weight distribution on iteration t
$D_t(i)$	weight distribution for class i on iteration t
N	number of classes
T	number of iterations
weak_learn(D_t)	weak learner algorithm with distribution D_t
e_t	weak_learn error on iteration t
β_t	error adjustment value on iteration t
Z_t	normalization constant for D_t
classifier	final boosted classifier
C	all classes

Figure 3.13 : AdaBoostM1 algorithm

Bootstrap aggregating, or "bagging" [breiman94], trains copies of a classifier algorithm on random bootstrap samples of the training set. When classifying a new document, each classifier votes and the aggregated votes determine the final classification. Bagging differs from boosting in the way that weight distributions are modified.

3.4.6 Other machine-learning techniques

Other machine-learning techniques, not employed by current recommender systems, include decision trees, neural networks, inductive logic learning and reinforcement learning. These techniques can be found in interface agents and other related technologies.

Decision trees build a tree-like decision structure based on the term weights in a training set. New vectors navigate the tree from the top, and when the bottom is reached the classification for that node used. Induction logic learns rules from a training set, which classify new instances based on term weights.

Neural networks learn patterns within the term vectors via a network of nodes and connections. Reinforcement learning adjusts weights based on the successful actions and is used to learn patterns of behaviour.

The next chapter describes the Quickstep recommender system, which a multi-class profile representation based on an ontology along with a boosted nearest neighbour machine-learning technique and time-decay profiling algorithm. Experimental work with Quickstep provides evidence to prove the first sub-hypothesis, which makes up part of the central hypothesis of this thesis.

Chapter 4 The Quickstep recommender system

Chapter summary
The Quickstep problem domain is presented.
An overview of the Quickstep system is detailed, and the empirical evaluation summarized.
Detailed descriptions of the approaches used by Quickstep are laid out.
Experimental set-up is described along with subject selection, experimental conditions and metrics recorded.
The experimental data is detailed and significant trends identified.
The trends seen are discussed and hypotheses offered for the effects seen. Comparison is made with other published data from similar systems.
Conclusions are drawn and evidence to support the first sub-hypothesis found.

Quickstep is an experimental recommender system. It addresses a real world problem and assesses the effectiveness of using an ontology within the profiling process. The overall effectiveness of this approach is measured and compared to other recommender systems within the literature. Appendix B contains details of the Quickstep implementation and a set of data flow diagrams detailing the system design.

The Quickstep system is described and evaluated in the K-CAP publication [middleton01b].

4.1 The Quickstep problem domain

As the trend to publish research papers on-line increases, researchers are increasingly using the web as their primary source of papers. Typically, researchers need to know about new papers in their field of interest, and older papers relating to their current work. In addition, a researcher's time is limited, so browsing competes with other tasks in the work place. It is this problem the Quickstep recommender system addresses.

Since researchers have their usual work to perform, unobtrusive monitoring methods are preferred because a researcher will be reluctant to use the system if it significantly interrupts normal workflow. Also, high recommendation accuracy is not critical as long as the system is deemed useful to them.

Evaluation of real world knowledge acquisition systems [shadbolt99] is both tricky and complex. A lot of evaluations are performed with user log data, simulating real user activity, or with standard benchmark collections that provide a basis for comparison with other systems. Although these evaluations are useful, especially for technique comparison, it is important to back them up with real world studies so we can see how the benchmark tests generalize to a real world setting.

This is why a real problem has been chosen upon which to evaluate the Quickstep recommender system.

4.2 Overview of the Quickstep system

Quickstep unobtrusively monitors user browsing behaviour via a web proxy server, logging each URL browsed during normal work activity. A machine-learning algorithm classifies browsed URLs overnight, and saves each classified paper in a central paper store. Explicit feedback and browsed topics form the basis of the interest profile for each user. Figure 4.1 shows an overview diagram of the Quickstep system.

Each day a set of recommendations is computed, based on correlations between user interest profiles and classified paper topics. Any feedback offered on these recommendations is recorded when the user looks at them.

Users can provide new examples of topics and correct paper classifications where appropriate. In this way the training set improves over time as more feedback is elicited from the users.

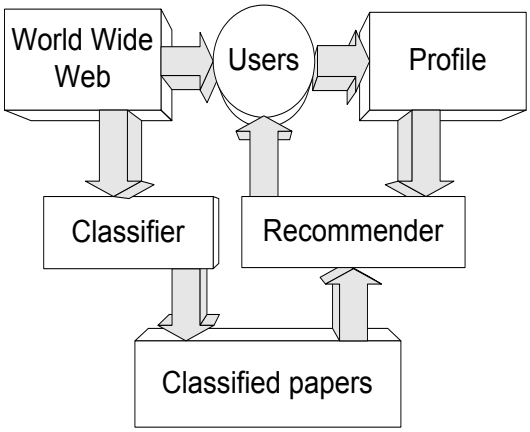


Figure 4.1 : The Quickstep system

4.3 Empirical evaluation

The current literature lacks many clear results showing the extent to which knowledge-based approaches assist real-world systems that have noisy data and differing user opinions. The evaluation compares the use of an ontology against a simple flat list, providing some empirical evidence as to the effectiveness of this knowledge-based approach.

Two experiments are described in detail later in this chapter. The first has 14 subjects, all using the Quickstep system for a period of 1.5 months. The second has 24 subjects, again over a period of 1.5 months.

Both experiments divide the subjects into two groups.

The first group uses a flat, extensible list of paper topics. Any new examples, added via explicit feedback, use this flat list to select from. The users are free to add to the list as needed.

The second group uses a fixed-size topic ontology based on the dmoz open directory project hierarchy [dmoz]. This ontology models the is-a relationships between computer science topics. The research topics used by the system are taken from this ontology. Interest profiles for the second group take into account the super classes of browsed topics, thus taking a knowledge-based approach to profile construction.

Performance metrics are measured over the duration of the trial, and the effectiveness of both groups compared.

4.4 The Quickstep system approach

Quickstep is a hybrid recommendation system, combining both content-based and collaborative filtering techniques. Since both web pages and user interests are dynamic in nature, catalogues, rule-bases and static user profiles would quickly become out of date. The recommender system approach is well suited to the problem, since it works with observed dynamic behaviour and will adjust to the changing interests of its users on a daily basis.

Asking users to provide explicit feedback on browsed papers would be very intrusive and interfere with the normal workflow of the researchers. Unobtrusive monitoring of web browsing was thus chosen to acquire positive examples of user interest. Optional explicit feedback is elicited when recommendations are made, both on topic interest and paper classification accuracy. However, since users choose when to review their recommendations, asking for explicit feedback at this point will not interrupt their normal workflow as they have chosen to spend time with the system anyway. Since many users will be using the system at once it is sensible to share user feedback and maintain a common pool of example papers provided by the users.

An initial training set of example papers is provided for each topic in the ontology to bootstrap the classifier. This training set is then allowed to grow over time as users provide examples of their own or correct the bootstrap examples. This labelled training set is well suited to supervised learning techniques, which require a prior set of classes on which to base a classification. A term vector representation is used to represent research papers, a common approach in machine-learning [mladenić99]. A term vector is a list of word weights, derived in this case from the frequency that words appear within the research paper main text.

A binary classification approach could have been used, with classes for “interesting” and “not interesting”. This would have led to profiles consisting of two term vectors, one representing the kind of thing the user is interested in, or positive examples, and the other what the user is not interested in, or negative examples. Recommendations would be those vectors that are most similar to the positive class vector and least

similar to the negative class vector. The binary case is the simplest class representation, and consequently produces the best classification results when compared with multi-class methods.

However, one problem with such a representation is that the explicit knowledge of topics in which the user is interested is lost, making it hard to benefit from any prior knowledge we may have about the academic domain. For Quickstep, a multi-class representation was chosen, with each class representing a research paper topic and class relationships held within an ontology. This allows profiles that consist of a human understandable list of topics, since each class represents a real world research topic. Quickstep's multi-class classifier assigns each new paper a class based on which class vector it is most similar to. Recommendations are selected from those papers classified as belonging to a topic of interest to a user.

The profile itself is computed from the correlation between browsed papers and paper topics. This correlation creates a topic interest history for each user, and a time-decay function is applied to compute the current topics of interest. A more complex profiling function, such as polynomial curve fitting or a machine-learning technique, would have trouble discriminating between the multiple interests people have since real world browsing behaviour data is both incomplete and noisy.

4.4.1 Research paper representation

Research papers are represented as term vectors, with term frequency / total number of terms used for a terms weight. To reduce the dimensionality of the vectors, term frequencies less than 2 are removed, standard Porter stemming [porter80] applied to remove word suffixes and the SMART [smart74] stop list used to remove common words such as "the". These measures are commonly used in information systems; [van rijsbergen79] and [harman86] provide a good discussion of these issues.

To give a rough idea of the size of the term vectors, 10-15,000 terms were used in the trials with training set sizes of about 200 vectors. Because of the training set size, further dimensionality reduction was not deemed necessary given the computing power and memory available. Had more dimensionality reduction been needed, term frequency-inverse document frequency (TF-IDF) weighting would have been useful,

with term weights below a threshold being removed, and latent semantic indexing (LSI) could also have been used.

4.4.2 Research paper classification

Since users can add to the training set over time and different users will pick different labels for specific examples, multiple labels must be supported. The classification requirements are thus for a multi-class learning algorithm learning from a multi-labelled training set. To learn from a training set, inductive learning is required. There are quite a few inductive learning techniques to choose from, including information theoretic ones (e.g. Rocchio classifier), neural networks (e.g. backpropagation), instance-based methods (e.g. nearest neighbour), rule learners (e.g. RIPPER), decision trees (e.g. C4.5) and probabilistic classifiers (e.g. naïve Bayes).

Multiple classifier techniques such as boosting exist as well, which can enhance the performance of individual classifiers.

After reviewing and testing many of the above options, a nearest neighbour technique was chosen. The nearest neighbour approach is well suited to the problem, since the training set must grow over time and consists of multi-class examples. Nearest neighbour algorithms also degrade well, with the next closest match being reported if the correct one fails to be found. The IBk algorithm [aha91] was chosen as it outperformed naïve Bayes and a J48 decision tree in informal tests. The boosting technique AdaBoostM1 [freund96] is also used, as it works well for multi-class problems if the boosted classifier is strong enough. In informal tests the boosting algorithm always improved the base classifier's performance.

Being a nearest neighbour algorithm, IBk stores instances of all example paper vectors in a vector space. To classify a new paper the vector distance from each example instance to the new paper vector is calculated, and the closest neighbours returned as the most likely classes. Inverse distance weighting is used to decrease the likelihood of choosing distant neighbours.

Example papers for non-leaf topics within the class hierarchy, which represent the more abstract topics, were selected from survey papers and review type papers of that topic. Papers not specific enough to be classified as a leaf node will thus generally end

up being classified as belonging to a more abstract topic. Survey papers were chosen as examples since they normally encompass a variety of sub-topics, and thus capture the range of concepts covered by the abstract topic.

AdaBoostM1 extends AdaBoost to handle multi-class cases since AdaBoost itself is a binary classifier. AdaBoostM1 repeatedly runs a weak learning algorithm, in this case the IBk classifier, for a number of iterations over various parts of the training set. The classifiers produced, each specialized for a particular class, are combined to form a single composite classifier at the end when all iterations have been executed.

4.4.3 Profiling algorithm

The profiling algorithm performs correlation between the paper topic classifications and user browsing logs. Whenever a research paper is browsed that has a classified topic, it accumulates an interest score for that topic. Explicit feedback on recommendations also accumulates interest values for topics. The current interest of a topic is computed using the inverse time weighting algorithm below, applied to the user feedback instances.

$$\text{Topic interest} = \sum_{n=1}^{\text{no of instances}} \text{Interest value}(n) \,/\, \text{days old}(n)$$

Event interest values	Paper browsed = 1 Recommendation followed = 2 Topic rated interesting = 10 Topic rated not interesting = -10
Interest value for super-class per instance	= 50% of sub-class

Figure 4.2 : Quickstep profiling algorithm

The profile for each user consists of a list of topics and the current interest values computed for them. The interest value weighting was chosen to provide sufficient weight for an explicit feedback instance to dominate the profile for about a week, but after a week to allow browsed URLs again to become dominant. In this way the profile will adapt to changing user interests as time progresses.

Profile = (<user>,<topic>,<topic interest value>)*

e.g. ((someone,hypertext,-2.4)
(someone,agents,6.5)
(someone,machine learning,1.33))

Figure 4.3 : Quickstep profile representation

If the user is using the ontology based set of topics, all super classes gain a share when a topic receives some interest. The immediate super class receives 50% of the main topic's value. The next super class receives 25% and so on until the most general topic in the is-a hierarchy is reached. In this way, general topics are included in the profile rather than just the most specific ones, producing a more rounded profile. A 50% value was chosen to reflect the increased uncertainty that occurs the further from the directly observed topic you go. This is how domain knowledge is used to improve the profiling process, allowing inference of unseen topics that are related to the specific topics browsed.

4.4.4 Recommendation algorithm

Recommendations are formulated from a correlation between the user's current topics of interest and the papers classified as belonging to those topics. A paper is only recommended if it does not appear in the user's browsed URL log, ensuring that all recommendations have not been seen before. For each user, the top three interesting topics are selected with 10 recommendations made in total (making a 4/3/3 split of recommendations). The top three interesting topics can come from the ontologies non-leaf nodes as well as leaf nodes. Papers are ranked in order of the recommendation confidence before being presented to the user.

Recommendation confidence =classification confidence *
topic interest value

Figure 4.4 : Quickstep recommendation algorithm

The classification confidence is computed from the AdaBoostM1 algorithm's class probability value for that paper (somewhere between 0 and 1). Since non-leaf abstract topics are classified against a training set of survey and review papers they will be more general in nature than a leaf node topic paper. The flat-list group does not have a hierarchy with non-leaf nodes, so the option of recommending an abstract non-leaf

node never arises. The existence of non-leaf nodes is the crucial difference between the flat list group and ontology group.

4.4.5 Research paper topic ontology

The research paper topic ontology is based on the dmoz [dmoz] taxonomy of computer science topics, and was chosen since it is freely available, has had a significant amount of development and is regularly maintenance. It is an is-a hierarchy of paper topics, up to 4 levels deep (e.g. an "interface agents" paper is-a "agents" paper). Pre-trial interviews generated some additional topics that were added to the dmoz taxonomy, and a review by two domain experts validated the ontology for correctness before use in the trials. Figure 4.5 shows a section of the research paper topic ontology.

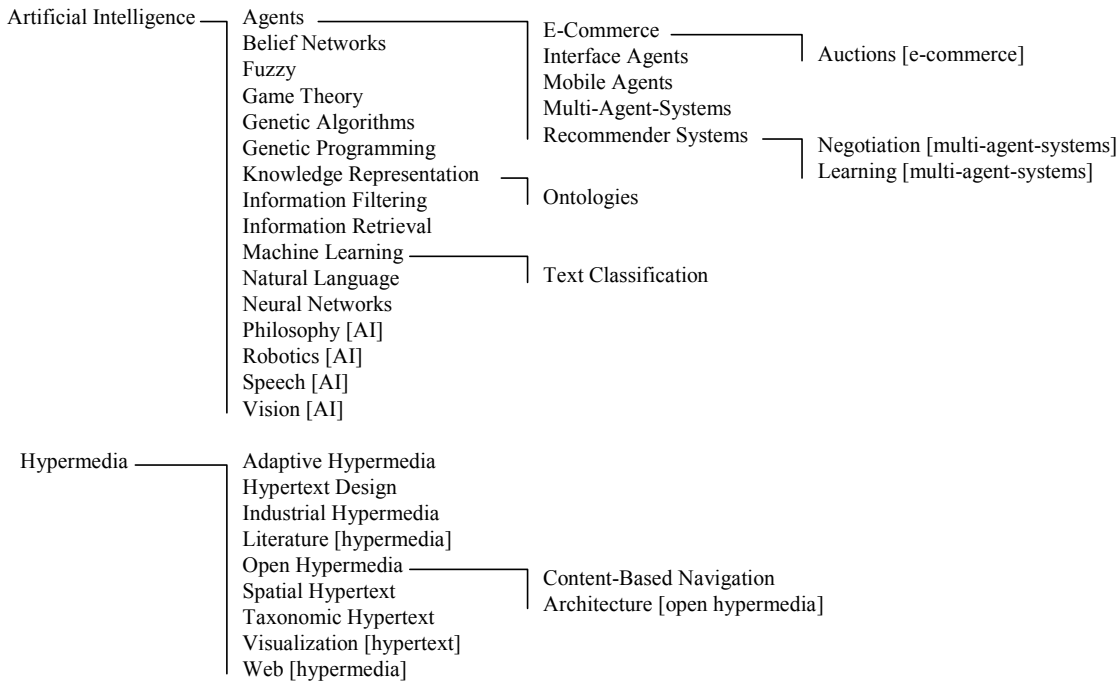


Figure 4.5 : Section of the Quickstep research paper ontology

A perfect ontology would tailor the granularity of topic size exactly to each users; if taken to an extreme this would require some sort of personal ontology per user and a mapping to shared example papers. To create such an ontology a full survey would be needed to elicit the types of research each user was interested in, and obtain some example papers for it. There is thus a balance to be made between the initial investment in knowledge engineering effort, which is multiplied by the number of

users, and what the system can learn by itself given enough time. There are also practical considerations, such as only maintaining a single ontology and the unwillingness of users to undergo a knowledge engineering exercise.

Ontologies themselves are discussed and defined in chapter 5, where the capabilities of an ontology is explored more fully.

4.4.6 Feedback and the quickstep interface

Recommendations are presented to the user via a browser web page. Figure 4.6 shows the web-based interface. The web page applet loads the current recommendation set and records any feedback the user provides. Research papers can be jumped to by clicking on them, and a new browser window will open and display the paper URL. If the user likes/dislikes the paper topic, the interest feedback combo-box allows "interested" or "not interested" to replace the default "no comment".

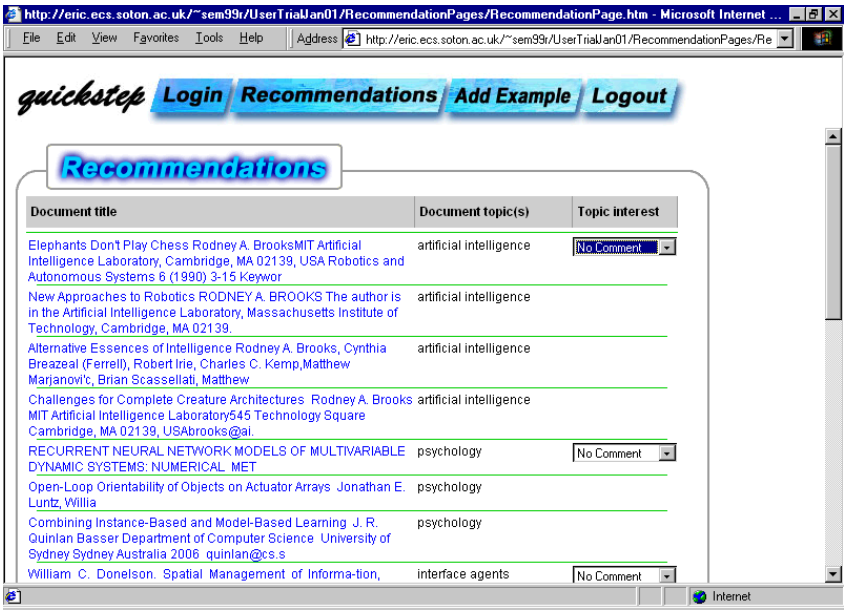


Figure 4.6 : Quickstep's web-based interface

The topic of the paper can be changed by clicking on the topic and selecting a new one from the popup list. Figure 4.7 shows this popup menu in use. The ontology group has a hierarchical popup menu, the flat list group has a single level popup menu.

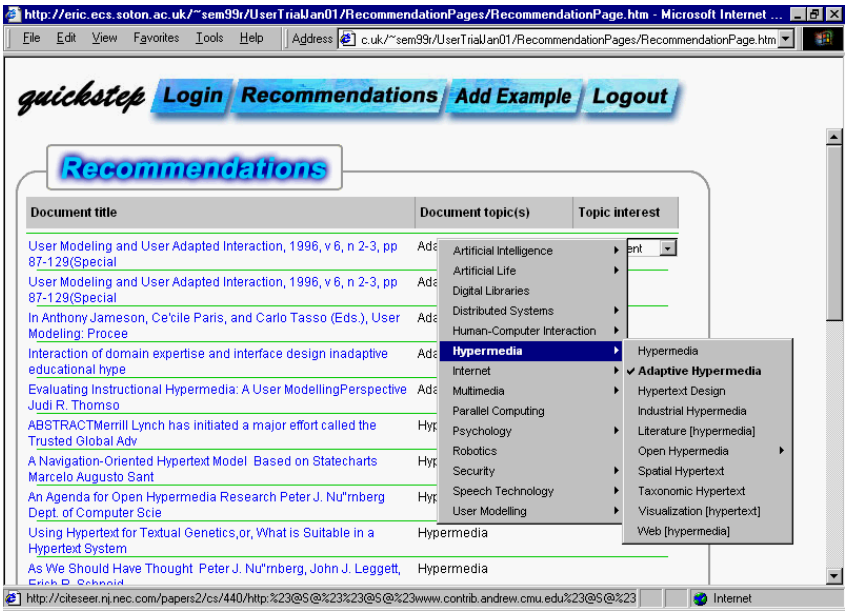


Figure 4.7 : Changing paper topics in Quickstep

New examples can be added via the interface, with users providing a paper URL and a topic label. These are added to the group's training set, allowing users to teach the system new topics or improve the classification of existing ones.

All feedback is stored in log files, ready for the profile builder which runs each day. The feedback logs are also used as the primary metric for evaluation. Interest feedback, topic corrections and jumps to recommended papers are all recorded and time stamped.

4.4.7 Design choices made in the Quickstep system

Since an increasing number of research papers are published in postscript and PDF format, it was decided to only deal with these formats (along with gzipped, zipped and Z compressed versions). The reason was to filter noisy HTML pages, of which only a fraction are research papers, and thus make the classification task easier. The drawback is research areas that publish primarily in HTML will be ignored.

The Weka machine-learning libraries [witten00] are used to implement the AdaBoostM1 classification algorithm and the IBk classifier.

Through informal empirical evaluation, 100 boosting iterations with 5% of the training set used per iteration proved best and was chosen for the trial. A k value of 5 was selected for the boosted IBk classifier.

The users were asked, in a pre-trial interview, to provide a few bookmarks to publication pages of important authors in their research areas of interest. Some of these bookmarks were pointed at digital libraries, a very useful source of freely available training examples. The bookmarks provided the Quickstep web crawler algorithm with somewhere to look initially. An initial set of papers was thus loaded into the system, making a pool of classified papers that could be recommended.

In addition to bookmarks, a manual bootstrap training set was created for each of the topics mentioned in the pre-trial interview. The use of a bootstrap training set reduces the burden on users to train the system before it becomes useful. Both the ontology group and the flat list group started the trials with an identical bootstrap training set, which was then allowed to diverge as the trial progressed.

To keep the trial simple (both to develop and analyse), feedback on the quality of individual papers was not elicited, only feedback on the interest of the paper topic to the user.

4.5 Experimental evaluation of Quickstep

Two experiments have been performed using the Quickstep system. Both compare the use of a flat, unstructured list of research paper topics to the use of a hierarchical is-a taxonomy. The profiling effectiveness is measured in addition to the overall usefulness of the system. In this way the benefit of using an ontological approach to profiling is evaluated. A binary comparison type experimental design was chosen for simplicity, and to reduce the number of potential causes for the effects seen.

4.5.1 Details of the two trials

Two trials were conducted to empirically assess both the overall effectiveness of the Quickstep recommender system and to quantify the effect of using an ontology in the profiling process.

The first trial used 14 subjects, consisting of researchers from Southampton University's Intelligence, Agents, Multimedia (IAM) research laboratory. A mixture of 2nd year postgraduates up to professors was taken, all using the Quickstep system for the duration of 1.5 months.

The second trial used 24 subjects, 14 from the first trial and 10 more 1st year postgraduates, and lasted for 1.5 months. Some minor interface improvements were made to make the feedback options less confusing.

A pre-trial interview obtained details from subjects such as area of interest and expected frequency of browser use. These details were used when deciding to which experimental groups subjects were to be allocated.

The purpose of both trials was to compare a group of users using an ontological approach to user profiling with a group of users using a flat, unstructured list. Subject selection for the groups tried to balance the groups as much as possible, evening out topics of interest, browser use and research experience in that order of importance. Both groups had the same number of subjects in them, 7 each for the first trial, 12 each for the second trial.

In the first trial, a bootstrap of 103 example papers covering 17 topics was used.

In the second trial, a bootstrap of 135 example papers covering 23 topics was used. The bootstrap training set for the second trial was expanded to include examples from the final training set of the first trial. The classified papers from the first trial were also kept, allowing a bigger initial collection of papers from which to recommend in the second trial. About five new classes were added to the ontology to cover those suggested by the flat list group in the first trial.

Both groups had their own separate training set of examples, which diverged from the bootstrap training set as the trial progressed. The classifier was run twice for each research paper, classifying once with the flat list group's training set and once with the ontology group's training set. The classifier algorithm was identical for both groups; only the training set changed.

The system interface used by both groups was identical, except for the popup menu for choosing paper topics. The ontology group had a hierarchical menu which used the ontology; the flat list group had a single level menu.

The system recorded each time the user declared an interest in a topic by selecting it "interesting" or "not interesting", jumped to a recommended paper or corrected the

topic of a recommended paper. These feedback events were date stamped and recorded in a log file for later analysis, along with a log of all recommendations made. Feedback recording was performed automatically by the system, whenever the subjects looked at their recommendations. The subjects were not aware of this recording process other than being told about it at the start of the trial.

A post-trial questionnaire was filled out after each trial, asking qualitative questions about the Quickstep system.

4.5.2 Experimental data

Since feedback is only collected when subjects check their recommendations, this data occurs at irregular times over the duration of the trial. Cumulative frequency of feedback events for each group is computed over the period of the trial, allowing trends to be seen as they develop during the trial. The total number of jumps and topics will differ between the two groups, hence the figures presented are normalized by dividing by the number of topics or recommendations up to that date. This avoids bias towards the group that provided feedback most frequently.

Figure 4.8 shows the topic interest feedback results. Topic interest feedback is where the user comments on a recommended topic, declaring it “interesting” or “not interesting”. If no feedback is offered, the result is “no comment”.

Topic interest feedback is an indication of the accuracy of the current profile. When a recommended topic is correct for a period of time, the user will tend to become content with it and stop rating it as “interesting”. On the other hand, an uninteresting topic is likely to always attract a “not interesting” rating. The absence of a “not interesting” rating is thus an indication of the acceptance of that topic, since “lazy” users will not go to the effort of looking at the recommendation page at all. Good topics are thus defined as either “no comment” or “interesting” topics. The cumulative frequency figures are presented as a ratio of the number of good topics to the total number of topics recommended. The not interesting ratio, bad topics, can be computed from these figures by subtracting the good topic value from 1.

The ontology groups have a 7% and 15% higher topic acceptance. In addition to this trend, the first trial ratios are about 10% lower than the second trial ratios.

Figure 4.9 shows the jump feedback results. Jump feedback is where the user jumps to a recommended paper by opening it via the web browser. Jumps are correlated with topic interest feedback, so a good jump is a jump to a paper on a good topic. Jump feedback is an indication of the quality of the recommendations being made as well as the accuracy of the profile. The cumulative frequency figures are presented as a ratio of the number of good jumps to total number of recommendations made.

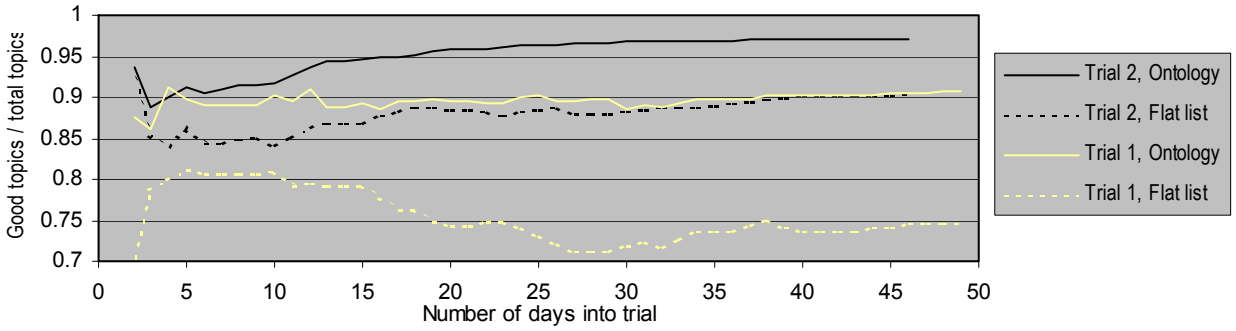


Figure 4.8 : Ratio of good topics / total topics

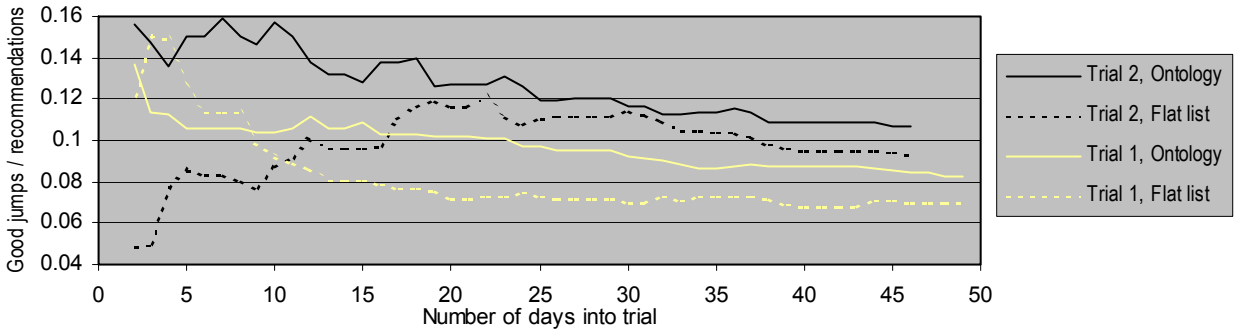


Figure 4.9 : Ratio of good jumps / total recommendations

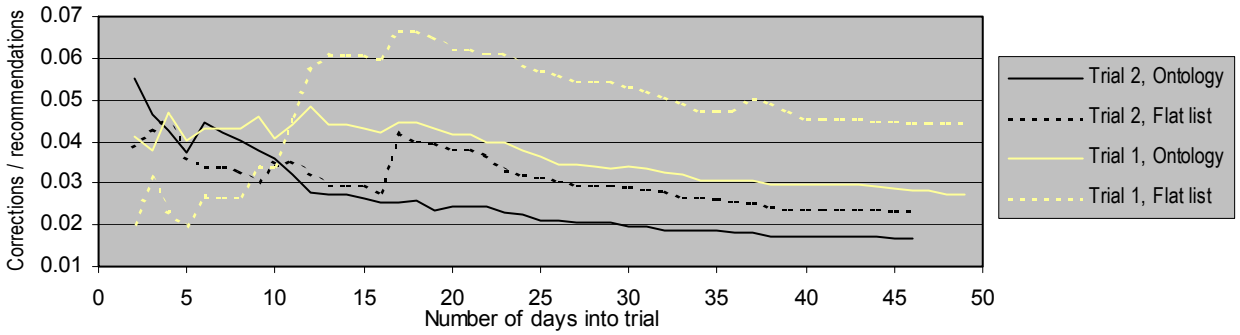


Figure 4.10 : Ratio of topic corrections / total recommendations

There is a small 1% improvement in good jumps by the ontology group. Both trials show between 8-10% of recommendations leading to good jumps.

Figure 4.10 shows the topic correction results. Topic corrections are where the user corrects the topic of a recommended paper by providing a new one. A topic correction will add to or modify a group's training set so that the classification for that group will improve. The number of corrections made is an indication of classifier accuracy. The cumulative frequency figures are presented as a ratio of the number of corrections to total number of recommended papers seen.

Although the flat list group has more corrections, the difference is only by about 1%. A clearer trend is for the flat list group corrections to peak around 10-20 days into the trial, and for both groups to improve as time goes on.

At the end of the first trial, the ontology group training set had 157 examples, and the flat list group had 162 examples. The paper repository had about 3000 classified research papers.

At the end of the second trial, the ontology group training set had 209 examples, and the flat list group had 212 examples. The paper repository had about 3500 classified research papers.

A cross-validation test was run on each group's final training set, to assess the precision and recall of the classifier using those training sets. The results are shown in table 4.1. The precision value is a measure of how many correctly classified documents there were as a proportion of the number classified. The recall value is a measure of how many documents were classified as a proportion of the total number of documents. Since the classifier does not use the ontological structure, the small variability seen in the precision and recall figures are due to the individual discriminating power of the example document terms found in each training set.

Group (trial)	Precision	Recall	Classes
Trial 1, Ontology	0.484	0.903	27
Trial 1, Flat list	0.52	1.0	25
Trial 2, Ontology	0.457	0.888	32
Trial 2, Flat list	0.456	0.972	32

Table 4.1 : Quickstep classifier recall and precision

4.5.3 Post-trial questionnaires

Each subject was given a post-trial questionnaire to fill out. The questionnaire asked users to select topics that were of interest during the trial from a list of all topics known to the system. Some 5-point scale questions were asked, to elicit qualitative information about the usefulness of the system. A free form comment section was also included so improvements could be suggested to the system.

The second trial's questionnaire replies are presented in table 4.2; some subjects felt unable to answer the questions so left some of them blank. The 5-point scale ranged from the lowest 1 to highest 5 value of answer, with textual comments associated with each value to guide the users.

Question	1	2	3	4	5	Mean
How much paper searching/reading did you do during the period of the trial?		11	8	2	2	2.78
How much did you use quickstep during the period of the trial?		10	7	6		2.83
Overall, were you interested in the topics recommended by quickstep?		4	11	7		3.14
Overall, how useful were the papers recommended by the quickstep system?		7	11	4		2.86
How accurate was quickstep at classifying papers?		1	13	8		3.32

Table 4.2 : Post-trial answers for Quickstep's second trial

The results indicate that the system recommended papers on topics that were fairly interesting but with only average usefulness. This is to be expected since the recommendation algorithm does not consider paper quality.

4.5.4 Discussion of trends seen in the experimental data

From the experimental data of both trials, several suggestive trends are apparent. The initial ratios of good topics were lower than the final ratios, reflecting the time it takes for enough log information to be accumulated to let the profiles settle down. The ontology users were 7-15% happier overall with the topics suggested to them.

A hypothesis for the ontology groups' apparently superior performance is that the is-a hierarchy produces a rounder, more complete profile by including general super class topics when a user browses a specific topic. This in turn helps the profiler to discover a broad range of interests, rather than just latching on to one correct topic.

The first trial showed fewer good topics than the second trial with a 10% difference seen by both groups. This is probably because of interface improvements made for the second trial, where the topic feedback interface was made less confusing. Subjects were sometimes rating interesting topics as not interesting if the paper quality was poor. As there are more poor quality papers than good quality ones, this introduced a bias to not interesting topic feedback resulting in a lower overall ratio.

About 10% of recommendations led to good jumps. Since ten recommendations were given to the users at a time, on average one good jump was made from each set of recommendations received. As with the topic feedback, the ontology group again was marginally superior but only by a 1% margin. This smaller difference is probably due to people having time to follow only one or two recommendations. Thus, although the ontology group has more good topics, only the top topic of the three recommended will really be looked at; the result is a smaller difference between the good jumps made and the good topics seen.

The flat list group has a poor correction / recommendation ratio 10-20 days into the trial. This is probably due to new topics being added to the system. Most new topics were added after the users became familiar with the system, and know which topics they feel are missing. The familiarization process appeared to take about ten days. The classification accuracy of these new topics is poor until enough examples have been entered, typically after another ten days.

The ontology group has about 1% fewer corrections for both trials. This small difference may indicate the utility of imposing a uniform conceptual model of paper topics on the subjects by using the common topic hierarchy. Classifying papers is a subjective process, and will surely be helped if people have similar ideas as to where topics fit in a groups overall classification scheme.

These preliminary results need to be extended to enable the application of more rigorous statistical analysis. Nevertheless, the trend in the data appears to be encouraging as to the utility of ontologies in recommender systems.

4.6 Comparison with other work in the literature

When compared with other published systems, the classification accuracy figures are similar, if on the low side, primarily because of the use of multi-class classification. Nearest neighbour systems such as NewsDude [billsus98] and Personal Webwatcher [mladenić96] report 60-90% classification accuracy based on binary classification. The higher figures tend to be seen with benchmark document collections, not real-world data. NewsWeeder [lang95] reports 40-60% classification accuracy using real user browsing data from two users over a period of time, so this would be the best comparison. If the number of classes classified is taken into consideration, Quickstep compares well. Multi-class classification is not normally applied to recommender systems, making direct comparison of other multi-class systems difficult.

This really highlights a problem in the recommender system literature today. There are virtually no experiments being performed with real users on realistic problems. As can be seen from the literature review in appendix A, most systems evaluate their performance by measuring classification accuracy using benchmark data or using one of the standard benchmark usage logs such as the movie lens dataset. While these metrics are good for comparison, they do not reveal how such systems work in practice, under the rigorous conditions of a real user trial. Running user trials is a long and expensive business, but such experiments are essential if the correlation between good benchmark performance and good real world performance is to be proved.

A comparison of the usefulness of our Quickstep to that of other systems was attempted, but the lack of published experimental data of this kind meant that only classification accuracy could be usefully compared. While the results found for

usefulness were not statistically significant, this was because of the attenuation that occurs when noisy, real world data is used for evaluation. While the exact size of the trends seen is not clear, the fact that they occurred in both trials is a significant indication of the benefits of an ontological approach to user profiling.

Chapter 2 and appendix A lists many recommender systems in the literature today. The systems most related to Quickstep are Entrée [burke00], which uses a knowledge base and case-based reasoning to recommend restaurant data, and RAAP [delgado98] that uses categories to represent user profiles with unshared individual training sets for each user. None of the reviewed systems use ontological categories to represent user profiles.

4.7 Conclusions from the Quickstep trials

Most recommender systems use a simple binary class approach, using a user profile of what is interesting or not interesting to the user. The Quickstep recommender system uses a multi-class approach, allowing a profile to be represented in terms of domain concepts, i.e. research paper topics, to be built. Multi-class classification is less accurate than binary classification, but allows class specific feedback and the use of domain knowledge, via an is-a hierarchy, to enhance the profiling process.

Two experiments were performed in a real work setting, using 14 and 24 subjects over a period of 1.5 months. The results suggest how using an ontology in the profiling process results in superior performance over using a flat list of topics. The ontology users tended to have "rounded" profiles, including more general topics of interest that were not directly suggested. This increased the accuracy of the profiles, and hence usefulness of the recommendations.

An informal result was seen in the nearest neighbour classifier's robustness. Even when classification was incorrect, 60% of the time in fact, the class chosen was normally in the correct area. For example, for an "interface agent" paper the classification would more likely be "agent" than "human computer interaction". The users liked this as it showed the system was at least making a reasonable attempt at classification, even if it was getting things wrong.

Although hard to compare, owing to the lack of a standard for reporting results, the overall recommender performance does appear to compare reasonably with other recommender systems in the literature.

This chapter thus provides the evidence to support the first sub-hypothesis, that the loss of information which occurs while representing a profile using an ontology is compensated for by the information gained by any inferences which are now possible. The next chapter examines the evidence to supports the second sub-hypothesis.

Chapter 5 Cold-start recommendation and ontology interest acquisition

Chapter summary
The cold-start and interest acquisition problems are defined.
Quickstep is introduced in the context of this integration.
OntoCoPI is introduced, a communities of practice identifier.
The integration of Quickstep and OntoCoPI is detailed.
An evaluation of the effectiveness of bootstrapping Quickstep is performed.
Conclusions are drawn and evidence to support the second sub-hypothesis found.

Recommender systems learn about user preferences over time, automatically finding things of similar interest. This reduces the burden of creating explicit queries. Recommender systems do, however, suffer from cold-start problems where no information is available early on upon which to base recommendations.

Semantic knowledge structures, such as ontologies, can provide valuable domain knowledge and user information. However, acquiring such knowledge and keeping it up to date is not a trivial task and user interests are particularly difficult to acquire and maintain.

This chapter investigates the synergy between a web-based research paper recommender system and an ontology containing information automatically extracted from departmental databases available on the web. The ontology is used to address the recommender system's cold-start problem. The recommender system addresses the ontology's interest-acquisition problem. An empirical evaluation of this approach is conducted and the performance of the integrated systems measured.

Evidence of enhanced cold-start performance in this chapter provided the evidence required to prove the second sub-hypothesis of this thesis.

5.1 Synergy between ontologies and recommender systems

Recommender systems [resnick97] learn about user preferences over time and automatically find things of similar interest, thus reducing the burden of creating

explicit queries. They track users dynamically as their interests change. However, such systems require an initial learning phase where behaviour information is built up to form a user profile. During this initial learning phase performance is often poor due to the lack of user information; this is known as the cold-start problem [maltz95].

There has been increasing interest in developing and using tools for creating annotated content and making it available over the semantic web. Ontologies are one such tool, used to maintain and provide access to specific knowledge repositories. Such sources could complement the behavioural information held within recommender systems, by providing some initial knowledge about users and their domains of interest. It should thus be possible to bootstrap the initial learning phase of a recommender system with such knowledge, easing the cold-start problem.

In return for any bootstrap information the recommender system could provide details of dynamic user interests to the ontology. This would reduce the effort involved in acquiring and maintaining knowledge of people's research interests. This chapter investigates the integration of Quickstep, a web-based recommender system, an ontology for the academic domain and OntoCoPI, a community of practice identifier that can pick out similar users.

5.1.1 The cold-start problem

One difficult problem commonly faced by recommender systems is the cold-start problem [maltz95], where recommendations are required for new items or users for whom little or no information has yet been acquired. Poor performance resulting from a cold-start can deter user uptake of a recommender system. This effect is thus self-destructive, since the recommender never achieves good performance because users never use it for long enough. Two types of cold-start problem are examined in this chapter.

The *new-system cold-start* problem is where there are no initial ratings by users, and hence no profiles of users. In this situation recommender systems have no basis on which to recommend, and hence perform very poorly.

The *new-user cold-start* problem is where the system has been running for a while and a set of user profiles and ratings exist, but no information is available about a new user. Recommender systems perform poorly in this situation too.

Collaborative recommender systems fail to help in cold-start situations, as they cannot discover similar user behaviour because there is not enough previously logged behaviour data upon which to base any correlations. Content-based and hybrid recommender systems perform a little better since they need just a few examples of user interest in order to find similar items.

No recommender system can cope alone with a totally cold-start, however, since even content-based recommenders require a small number of examples on which to base recommendations. The work in this chapter links together a recommender system and an ontology to address this problem. The ontology can provide a variety of information on users and their publications. Publications provide important information about what interests a user has had in the past, and so provide a basis upon which to create initial profiles that can address the new-system cold-start problem. Personnel records allow similar users to be identified. This will address the new-user cold-start problem by providing a set of similar users on which to base a new-user profile.

5.1.2 Ontologies

An ontology is a conceptualisation of a domain into a human-understandable, but machine-readable format consisting of entities, attributes, relationships and axioms [guarino95]. Ontologies can provide a rich conceptualisation of the working domain of an organisation, representing the main concepts and relationships of the work activities. These relationships could represent isolated information such as an employee's home phone number, or they could represent an activity such as authoring a document, or attending a conference.

Throughout this thesis the term ontology is used to refer to the classification structure and instances within the knowledge base.

The ontology [akt-ontology] used in this work is designed to represent the academic domain, and was developed by the Southampton's Advanced Knowledge

Technologies (AKT) team. It models people, projects, papers, events and research interests. The ontology itself is implemented in Protégé 2000 [eriksson99], a graphical tool for developing knowledge-based systems. It is populated with information extracted automatically from a departmental personnel database and publication database. The ontology consists of around 80 classes, 40 slots, over 13000 instances and is focused on people, projects and publications.

5.1.3 The interest-acquisition problem

People's areas of expertise and interests are an important type of knowledge for many applications, for example expert finders [dunlop00]. Semantic web technology can be a good source of such information, but usually requires substantial maintenance to keep the web pages up-to-date. Since web page maintenance is time consuming, the majority of web pages receive little maintenance, holding information that does not age quickly. Since interests and areas of expertise will often change, they are not often held within web pages. It is thus particularly difficult for an ontology to acquire such information; this is the *interest-acquisition* problem.

Many existing systems force users to perform self-assessment to gather such information, but this has numerous disadvantages [becerra-fernandez00]. Lotus have developed a system that monitors user interaction with a document to capture interests and expertise [lotus01]. Their system does not, however, consider the online documents that users browse.

This chapter investigates linking an ontology with a recommender system to help overcome the interest acquisition problem. The recommender system will regularly provide the ontology with interest profiles for users, obtained by monitoring user web browsing and analysing feedback on recommended research papers.

5.2 OntoCoPI

The Ontology-based Communities of Practice Identifier (OntoCoPI) [alani02] is an experimental system developed by the AKT project, which uses the Southampton populated AKT ontology to help identify communities of practice (CoP). The community of practice of a person is taken here to be the closest group of people, based on specific features they have in common with that given person. A community

of practice is thus an informal group of people who share some common interest in a particular practice [brown00] [wenger00]. Workplace communities of practice improve organisational performance by maintaining implicit knowledge, helping the spread of new ideas and solutions, acting as a focus for innovation and driving organisational strategy.

Identifying communities of practice is an essential first step to understanding the knowledge resources of an organization [wenger99]. Organisations can bring the right people together to help the identified communities of practice to flourish and expand, for example by providing them with appropriate infrastructure and giving them support and recognition. However, community of practice identification is currently a resource-heavy process largely based on interviews, mainly because of the informal nature of such community structures that are normally hidden within and across organisations.

OntoCoPI is a tool that uses ontology-based network analysis to support the task of community of practice identification. A breadth-first spreading activation algorithm is applied by OntoCoPI to crawl the ontology network of instances and relationships to extract patterns of certain relations between entities relating to a community of practice. The crawl can be limited to a given set of ontology relationships. These relationships can be traced to find specific information, such as who attended the same events, who co-authored papers and who are members of the same project or organisation. Communities of practice are based on informal sets of relationships while ontologies are normally made up of formal relationships. The hypothesis underlying OntoCoPI is that some informal relationships can be inferred from the presence of formal ones. For instance, if A and B have no formal relationships, but they have both authored papers with C, then that could indicate a shared interest.

One of the advantages of using an ontology to identify communities of practice, rather than other traditional information networks [albert02], is that relationships can be selected according to their semantics and can have different weights to reflect relative importance. For example the relations of document authorship and project membership can be selected if it is required to identify communities of practice based on publications and project work. OntoCoPI allows manual selection of relationships or automatic selection based on the frequency of relationship use within the knowledge base. Selecting the right relationships and weights is an experimental

process that is dependent on the ontology structure, the type and amount of information in the ontology, and the type of community of practice required.

When working with a new community of practice, some experiments will be needed to see which relationships are relevant to the desired community of practice, and how to set relative weights. In the experiments described in this chapter, certain relationships were selected manually and weighted based on our preferences. The most highly weighted relationships were supervision, project membership and authorship relations. The AKT ontology research topic labels were manually altered to match those labels used by the Quickstep system to avoid ontological label mapping issues. Further trials would be needed to determine the most effective relationship weightings.

5.3 Integrating Quickstep, OntoCoPI and the AKT ontology

This chapter integrates the ontology, OntoCoPI and Quickstep recommender system to provide a solution to both the cold-start problem and interest-acquisition problem. Figure 5.1 shows the experimental systems after integration.

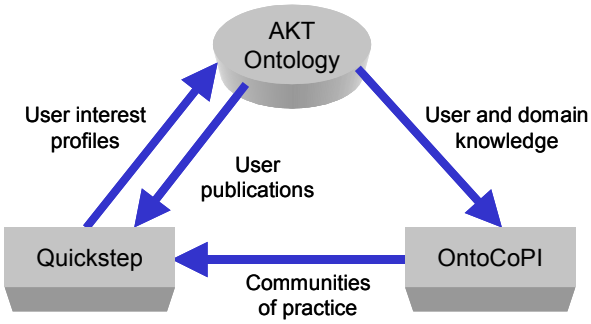


Figure 5.1 : Ontology and recommender system integration

Upon start-up, the ontology provides the recommender system with an initial set of publications for each of its registered users. Each user's known publications are then correlated with the recommender systems classified paper database, and a set of historical interests compiled for that user. These historical interests form the basis of an initial profile, overcoming the new-system cold-start problem. Figure 5.2 details the initial profile algorithm. As per the Quickstep profiling algorithm, fractional interest in topic super-classes are inferred when a specific topic is added.

$$\text{topic interest}(t) = \sum_{n=1}^{\text{publications belonging to class t}} 1 / \text{publication age}(n)$$

new-system initial profile = (t, topic interest(t))*

t = <research paper topic>

Figure 5.2 : New-system initial profile algorithm

When a new user is added to the recommender system after an initial period of use, the ontology provides the historical publication list of the new user and the OntoCoPI system provides a ranked list of similar users. The initial profile of the new user is formed from a correlation between historical publications and any similar user profiles. This algorithm is detailed in figure 5.3, and addresses the new-user cold-start problem.

$$\text{topic interest}(t) = \frac{\gamma}{N_{similar}} \sum_{u=1}^{N_{similar}} \text{profile interest}(u,t) + \sum_{n=1}^{N_{pubs\ t}} 1 / \text{publication age}(n)$$

profile interest(u,t) = interest of user u in topic t * CoP confidence
new-user initial profile = (t, topic interest(t))*

t = research paper topic
u = user
γ = weighting constant >= 0
$N_{similar}$ = number of similar users
$N_{pubs\ t}$ = number of publications belonging to class t
CoP confidence = Communities of practice confidence

Figure 5.3 : New-user initial profile algorithm

While only publications were directly mapped, the bootstrapping functions could be expanded to include other ontological concepts such as research areas. This would give access to specific user communities of practice, rather than ones dynamically generated by network analysis, and would probably have a higher confidence value.

The task of populating and maintaining the ontology of user research interests is left to the recommender system. The recommender system compiles user profiles on a daily basis, and these profiles are added into the ontology when ready. Figure 5.4 details the structure of these profiles. In this way up-to-date interests are maintained,

providing a solution to the interest acquisition problem. The interest data is used alongside the more static information within the ontology to improve the accuracy of the OntoCoPI system.

user profile = (topic, interest)*
topic = <research topic>
interest = <interest value>

Figure 5.4 : Daily profiles sent to the AKT ontology

The issue of mapping the Quickstep's research topic classes to those in the AKT ontology was avoided by changing the AKT ontology class names to match the Quickstep ontology class names. A more scalable solution would need to employ a set of mappings between ontological concepts. If more than one external ontology was to be integrated then it would make sense to build a specific module within the Quickstep recommender system to manage these mappings, and use a soft-coded description of each mappings, in a language like XML, for easy maintenance.

5.4 Example of integrated system operation

When the Quickstep recommender system is first initialised, it retrieves a list of people and their publication URLs from the ontology. Quickstep analyses these publications and classifies them according to the research topic hierarchy in the ontology. Paper topics are associated with their date of publication, and the 'new-system initial profile' algorithm used to compute a set of initial profiles for each user.

Tables 5.1 and 5.2 show an example of this for the user Nigel Shadbolt. His publications are analysed and a set of topics and dates formulated. The 'new-system initial profile' algorithm then computes the interest values for each topic. For example, 'Knowledge Acquisition' has one publication two years old (round up) so its value is 1.0 / 2 = 0.5.

Publication	*Date*	*Topic*
Capturing Knowledge of User Preferences: ontologies on recommender systems	2001	Recommender systems
Knowledge Technologies	2001	Knowledge Technology
The Use of Ontologies for Knowledge Acquisition	2001	Ontology
Certifying KBSs: Using CommonKADS to Provide Supporting Evidence for Fitness for Purpose of KBSs	2000	Knowledge Management
Extracting Focused Knowledge from the Semantic Web	2000	Knowledge Acquisition
Knowledge Engineering and Management	2000	Knowledge Management
…		

Table 5.1 : Publication list for Shadbolt

Topic	Interest
Knowledge Technology\Knowledge Management	1.5
Knowledge Technology\Ontology	1.0
AI\Agents\Recommender Systems	1.0
Knowledge Technology\Knowledge Acquisition	0.5
…	

Table 5.2 : Example of new-system profile for Shadbolt

At a later stage, after Quickstep has been running for a while, a new user registers with email address sem99r@ecs.soton.ac.uk. OntoCoPI identifies this email account as that of Stuart Middleton, a PhD candidate within the department, and returns the ranked and normalised communities of practice list displayed in table 5.3. This communities of practice list is identified using relations on conference attendance, supervision, authorship, research interest and project membership, using the weights 0.4, 0.7, 0.3, 0.8, and 0.5 respectively. De Roure was found to be the closest person as he is Middleton's supervisor, and has one joint publication co-authored with Middleton and Shadbolt. The people with 0.82 values are other supervisees of De Roure. Alani attended the same conference as Middleton went to in 2001.

Person	Relevance value	Person	Relevance value
DeRoure	1.0	Alani	0.47
Revill	0.82	Shadbolt	0.46
Beales	0.82		

Table 5.3 : OntoCoPI results for Middleton

The communities of practice list is then sent to Quickstep, which searches for matching user profiles. These profiles will be more accurate and up to date than those initially created profiles based on publications. Quickstep manages to find the profiles shown in table 5.4 in its logs.

Person	Topic	Interest
DeRoure	AI\Distributed Systems	1.2
	AI\Agents\Recommender Systems …	0.73
Revill	AI\Agents\Mobile Agents	1.0
	AI\Agents\Recommender Systems …	0.4
Beals	Knowledge Technology\Knowledge Devices	0.9
	AI\Agents\Mobile Agents …	0.87
Alani	Knowledge Technology\Ontology	1.8
	Knowledge Technology\Knowledge Management\ CoP …	0.7
Shadbolt	Knowledge Technology\Knowledge Management	1.5
	AI\Agents\Recommender Systems …	1.0

Table 5.4 : Profiles of similar people to Middleton

These profiles are merged to create a profile for the new user, Middleton, using the 'new-user initial profile' algorithm with a γ value of 2.5. For example, Middleton has a publication on 'Recommender Systems' that is one year old and De Roure, Revill and Shadbolt have an interest in 'Recommender Systems' – this topic's value is therefore $1/1 + 2.5/5 * (1.0*0.73+0.82*0.4+0.46*1.0) = 1.76$. Table 5.5 shows the resulting profile.

Topic	Interest
AI\Agents\Recommender Systems	1.76
AI\Agents\Mobile Agents	0.77
AI\Distributed Systems	0.6
Knowledge Technology\Ontology	0.42
Knowledge Technology\Knowledge Devices	0.37
Knowledge Technology\Knowledge Management	0.35
Knowledge Technology\Knowledge Management\ CoP	0.16
…	

Table 5.5 : New-user profile for Middleton

Every day Quickstep's profiles are updated and automatically fed back to the ontology, where they are used to populate the research interest relationships of the relevant people.

5.5 Empirical evaluation of the integrated system

In order to evaluate the effect both the new-system and new-user initial profiling algorithms have on the integrated system, an experiment was conducted based around the browsing behaviour logs obtained from the Quickstep [middleton01b] user trials. The algorithms previously described are used, as per the example in the previous section, and the average performance for all users calculated.

5.5.1 Experimental approach

Users were selected from the Quickstep trials had entries within the departmental publication database. Nine users were selected in total, with each user typically having one or two publications.

The URL browsing logs of these users, extracted from three months of browsing behaviour recorded during the Quickstep trials, were then broken up into weekly log entries. Seven weeks of browsing behaviour were taken from the start of the Quickstep trials, and an empty log created to simulate the very start of the trial.

Eight iterations of the integrated system were thus run, the first simulating the start of the trial and others simulating the following weeks 1 to 7. Interest profiles were recorded after each iteration. Two complete runs were made, one with the 'new-system initial profiling' algorithm and one without. The control run without the 'new-system initial profiling' algorithm started with blank profiles for each of its users.

An additional iteration was run to evaluate the effectiveness of the 'new-user initial profile' algorithm. The communities of practice for each user were taken, based on data from week 7, and used with the 'new-user initial profile' algorithm to compute initial profiles for each user as if they were being entered onto the system during the trial. These profiles were recorded. Because an early prototype version of OntoCoPI was used, communities of practice confidence values were not available; the confidence values were thus all of value 1 throughout this experiment.

In order to evaluate the algorithms effect on the cold-start problem, all recorded profiles were compared to the benchmark week 7 profile. This allowed measurement of how quickly profiles converge to the stable state existing after a reasonable amount

of behaviour data has been accumulated. The quicker the profiles move to this state the quicker they will have overcome the cold-start.

Week 7 was chosen as the cut-off point of our analysis since after about 7 weeks of use the behaviour data gathered by Quickstep will dominate the user profiles. The effects of bootstrapping beyond this point would not be significant. If the system were run beyond week 7 it would simply continually adjust the profiles to the behaviour logged each week.

5.6 Experimental results

Two measurements were made when comparing profiles to the benchmark week 7 profile. The first, profile precision, measures how many topics were mentioned in both the current profile and benchmark profile. Profile precision is an indication of how quickly the profile is converging to the final state, and thus how quickly the effects of the cold-start are overcome. The second, profile error rate, measures how many topics appeared in the current profile that did not appear within the benchmark profile. Profile error rate is an indication of the errors introduced by the two bootstrapping algorithms. Figure 5.5 describes these metrics.

It should be noted that the absolute precision and error rate of the profiles are not measured – only the relative precision and error rate compared to the week 7 steady state profiles. Measuring absolute profile accuracy is a very subjective matter, and is not attempted here; only how quickly profiles reach their steady states is measured. A more complete evaluation of Quickstep's overall profiling and recommendation performance can be found in [middleton01b].

$$\text{profile precision} = \frac{1}{N_{users}} \sum_{user=1}^{N_{users}} \frac{N_{correct}}{N_{correct} + N_{missing}}$$

$$\text{profile error rate} = \frac{1}{N_{users}} \sum_{user=1}^{N_{users}} \frac{N_{incorrect}}{N_{correct} + N_{incorrect} + N_{missing}}$$

$N_{correct}$	Number of user topics that appear in current profile and benchmark profile
$N_{missing}$	Number of user topics that appear in benchmark profile but not in current profile
$N_{incorrect}$	Number of user topics that appear in current profile but not in benchmark profile
N_{users}	Total number of users

Figure 5.5 : Evaluation metrics

The results of our experimental runs are detailed in figures 5.6 and 5.7. The new-user results consist of a single iteration, so appear on the graphs as a single point.

At the start, week 0, no browsing behaviour log data is available to the system so the profiles without bootstrapping are empty. The new-system algorithm, however, can bootstrap the initial user profiles and achieves a reasonable precision of 0.35 and a low error rate of 0.06. It was found that the new-system profiles accurately captured interests users had a year or so ago, but tended to miss current interests. This is because publications are generally not available for up-to-date interests.

As expected, once the weekly behaviour logs become available to the system the profiles adjust accordingly, moving away from the initial bootstrapping. On week 7 the profiles converge to the benchmark profile.

The new-user algorithm result show a more dramatic increase in precision to 0.84, but comes at the price of a significant error rate of 0.55. The profiles produced by the new-user algorithm tended to be very inclusive, taking the set of similar user interests and producing a union of these interests. While this captures many of the new user's real interests, it also included a large number of interests not relevant to the new user but which were interesting to the people similar to the new user.

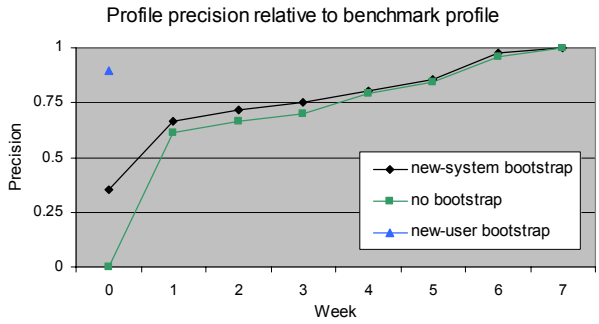


Figure 5.6 : Profile precision

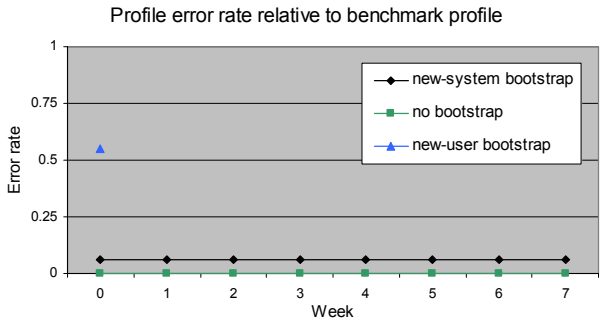


Figure 5.7 : Profile error rate

Since error rate is measured relative to the final benchmark profile of week 7, all the topics seen in the behaviour logs will be present within the benchmark profile. Incorrect topics must thus come from another source – in this case the bootstrapping on week 0. This causes error rates to be constant over the 7 weeks, since the incorrect topics introduced on week 0 remain for all subsequent weeks.

5.7 Conclusions

Cold-starts in recommender systems and interest acquisition in ontologies are significant problems. If initial recommendations are inaccurate, user confidence in the recommender system may drop with the result that not enough usage data is gathered to overcome the cold-start. With regard to ontologies, up-to-date interests are not generally available from periodically updated information sources such as web pages, personnel records or publication databases.

The integration of the Quickstep recommender system, AKT ontology and OntoCoPI system has demonstrated one approach to reduce both the cold-start and interest-acquisition problems. The practical work suggests that using an ontology to bootstrap

user profiles can significantly reduce the impact of the recommender system cold-start problem. It is particularly useful for the new-system cold-start problem, where the alternative is to start with no information at all. Regularly feeding the recommender system's interest profiles back to the ontology also clearly assists in the acquisition of up-to-date interests. A number of issues have, however, arisen from the integration.

The new-system algorithm produced profiles with a low error rate and a reasonable precision of 0.35. This reflects that previous publications are a good indication of user's current interests, and so can produce a good starting point for a bootstrap profile. Where the new-system algorithm fails is for more recent interests, which make up the remaining 65% of the topics in the final benchmark profile. To discover these more recent interests, it is possible that the new-system algorithm could be extended to take some of the other information available within the ontology into account, such as the projects a user is working on. To what degree these relationships will help is difficult to predict however, since the ontology itself has great difficulty in acquiring knowledge of recent interests.

For the purposes of this experiment, those users who had some entries within the university's on-line publication database were selected. There were some users who had not entered their publications into this database or who have yet to publish their work. For these users there is little information within the ontology, making any new-system initial profiles of little use. In a larger scale system, more sources of information would be needed from the ontology to build the new-system profiles. This would allow some redundancy, and hence improve robustness in the realistic situation where information is only sparsely available.

The community of practice for a user was found not to be always relevant based on our selection of relationships and weights. For example, David De Roure supervises Stuart Middleton, but he also supervises a lot of other students interested in mobile agents. These topics are not relevant to Stuart, which raises the question of how relevant the supervision relationship is to our requirements, and how best to weight such a relationship. The prototype version of OntoCoPI used in this experiment contained relationship weights biased towards this supervision relationship. Further experiments are needed to identify the most relevant settings for community of practice identification. The accuracy of our communities of practice are also linked to

the accuracy of the research interest information as identified by the recommender system.

The new-user algorithm achieved good precision of 0.84 at the expense of a significant 0.55 error rate. This was because both the communities of practice generated for users were not always precise, and because the new-user algorithm included all interests from the similar users. An improvement would be to only use those interests held by the majority of the people within a community of practice. This would exclude some of the less common interests that would otherwise be included into the new-user profile.

The new-user initial profile algorithm uses the constant γ, which determines the proportional significance of previous publications and similar users. Factors such as the availability of relationship data within the ontology and quality of the publication database will affect the choice of value for γ. A value of 2.5 was used, but empirical evaluation would be needed to determine the best value.

There is an issue as to how best to calculate the "semantic distance" between topics within the is-a hierarchy. The simplifying assumption that all is-a links have equal relevance is used here, but the exact relevance will depend on each topic in question. If individual weightings were allowed for each topic, a method for determination of these weights would have to be considered. Alternatively the is-a hierarchy could be carefully constructed to ensure equal semantic distance.

A positive feedback loop exists between the recommender system and ontology, making data incest a potential problem. For new users there are no initial interest entries within the ontology, so new user profiles are not incestuous. If the recommender system were to use the communities of practice for more than just initial profiles, however, a self-confirming loop would exist; the recommender system would derived its interests from the communities of practice, which would derive its interests from the recommender system, and so on in an incestuous loop. However, if the ontology were to obtain research interest information from another source then the interests could be used to refine the list of users in a community of practice further, and thus increase the accuracy of the bootstrapping algorithms.

Finally, a question still remains as to just how good an initial profile must be to fully overcome the effects of the cold-start problem. If initial recommendations are poor users will not use the recommender system and hence it will never get a chance to improve. It has been shown that improvements can be made to initial profiles, but further empirical evaluation would be needed to evaluate exactly how much improvement is needed before the system is "good enough" for users to give it a chance.

The bootstrapping used in this chapter was made possible because user profiles were represented using an ontology. Knowledge held within the ontology was thus able to provide the information required for initial bootstrapping. The benefits seen in using such knowledge thus provide the evidence required to support the second sub-hypothesis. Chapter 6 provides evidence to support the third sub-hypothesis, showing a further way in which an ontological representation of user profiles can offer benefits to a recommender system.

Chapter 6 The Foxtrot recommender system

Chapter summary
The Foxtrot problem domain is presented.
An overview of the Foxtrot system is detailed, and the empirical evaluation summarized.
Detailed descriptions of the approaches used by Foxtrot are laid out.
Experimental set-up is described along with subject selection, experimental conditions and the metrics recorded.
The experimental data is detailed and significant trends identified.
The trends seen are discussed and hypotheses offered for the effects seen. Comparison is made with other published data from similar systems.
Conclusions are drawn and evidence to support the third sub-hypothesis found.

This chapter investigates the Foxtrot recommender system, an evolution of the Quickstep system described in chapter 4. It addresses the problem domain of recommending academic research papers, as does Quickstep, but tackles a larger range of research topics and provides a service over a much longer time. Several enhancements are made to the interface, classification and recommendation processes and the recommendation facility is built upon a database search facility. Profile visualization is explored as an additional benefit of representing profiles in ontological terms, hence providing the evidence to support the third sub-hypothesis. An evaluation is carried out and both recommendation and profiling effectiveness assessed and compared to other recommender systems within the literature.

Appendix C contains details of the Foxtrot implementation and a set of data flow diagrams detailing the system design.

6.1 The Foxtrot problem domain

Foxtrot addresses the problem of recommending on-line research papers to over 300 computer science staff and students at the University of Southampton for a full academic year. The first user requirement is thus for a system that can search a research paper database and return on-line papers, and the second requirement for a

recommendation facility that can help suggest useful papers missed by explicit searching. Unobtrusive monitoring methods are preferred because researchers have their normal work to perform and would not welcome interruptions from a new system. Very high accuracy on recommendations is not required since users will have the option to simply ignore poor recommendations.

6.2 Overview of the Foxtrot system

The Foxtrot recommender system is a hybrid recommender system and searchable paper database. Collaborative and content-based recommendation is supported, in addition to direct database searching. Figure 6.1 shows an overview of the Foxtrot architecture. Foxtrot is an evolution of the Quickstep [middleton01b] recommender system, described in chapter 4.

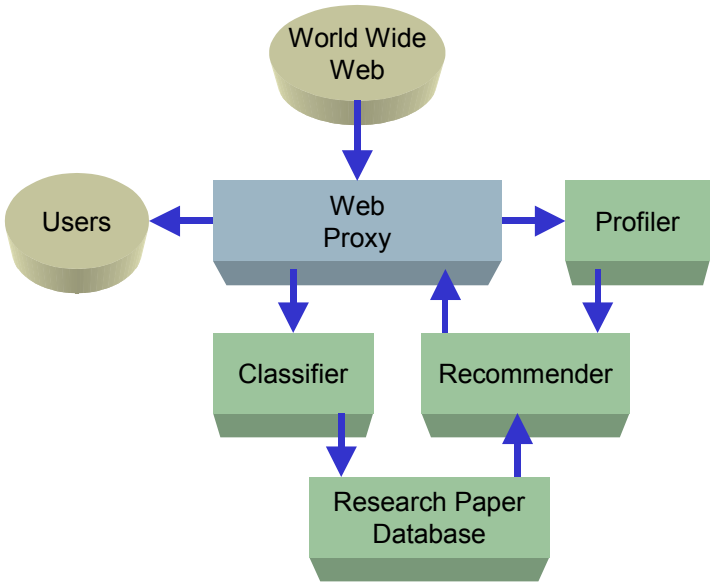


Figure 6.1 : Foxtrot overview

A web proxy is used to monitor unobtrusively each user's web browsing, adding new research papers to the central database as users discover them. The research paper database thus acts as a pool of shared knowledge, available to all users via search and recommendation. The database of research papers is classified using a research paper topic ontology and a set of training examples for each topic.

Recorded web browsing and relevance feedback elicited from users is used to compute daily profiles of users' research interests. Interest profiles are represented in ontological terms, allowing other interests to be inferred that go beyond that just seen

from directly observed behaviour. These interest profiles are visualized to allow elicitation of direct profile feedback, thus providing an additional source of information from which profiles can be computed.

Recommendations are compiled daily and suggested when a user goes to the Foxtrot web page or receives a weekly email. Collaborative filtering is used to compute the recommendations, using only the current topics of interests in each user's content-based profile.

6.3 Empirical evaluation

An empirical evaluation of the Foxtrot system was conducted with the computer science staff and students of the University of Southampton, over the period of an academic year. The aim of the evaluation was to assess the benefits of using direct profile visualization and feedback and to assess the overall effectiveness of the recommender system.

Subjects registered to use the system were randomly divided into two groups. The first group was allowed to see their profile and provide direct profile feedback, the second group was not allowed. Both groups had full access to the system and could provide traditional relevance feedback on the recommended papers and search results found when using the system.

Performance metrics were measured over the duration of the trial, and the effectiveness of both groups compared.

6.4 Interesting lessons from the Quickstep system

Before the Foxtrot recommender system design was finalized, a number of issues that were raised with the Quickstep system were addressed. These mainly came from the implementation of the interface, and the way researchers appeared to prefer to work.

Firstly there was a clear need for a searchable database in addition to the recommendation web page interface. Researchers often wanted to find a specific paper, or search for a paper on a specific topic, and did this via a normal search engine such as Google; they did not want to wait for the recommender to eventually suggest the paper. Interestingly, the Quickstep database of research papers would have been

an ideal place to look. Therefore it seemed useful to allow users to enter immediate search queries for the papers they know they need. The recommendation facility could then be used in conjunction to normal searching, recommending extra papers that a user might not have realized was out there or not bothered to find.

In addition to a search facility, it became clear that there was a lot to be gained from a collaborative approach in addition to a content-based approach to recommendation. Since the research topics were quite broad in nature, there were often cases when the system had to choose from several papers on the same topic; a collaborative filter here would allow the system to recommend papers on a topic which other users liked before papers that were untried.

Users tended to have difficulty with the concept of interest feedback. Some users thought this was for papers they liked and others for good quality papers. The solution to this semantic dilemma was to introduce an explicit quality rating in addition to the interest rating of the Quickstep system. This removed completely the ambiguity.

6.5 The Foxtrot System Approach

Two approaches are taken to help researchers effectively find the papers they need.

In the case where a researcher is looking for a specific paper, a search facility can access the Foxtrot research paper database. This allows researchers to enter search keywords and search the title, content and classification of papers in the database. A search facility is well suited to this problem since computer science researchers are normally experienced in using the web and formulating useful search queries.

In the case where the researcher is interested in keeping up-to-date in a research area, a recommendation facility has been built to suggest previously unseen papers that might be of interest. A recommender system approach is useful here because user interests will change over time and hence need to be continuously monitored. The recommender regularly generates user profiles from observed behaviour and uses this as the basis for recommending research papers.

Combining both the search engine and recommender system approaches allows the advantages of both to be combined and hence provides a more effective tool for researchers.

Regularly asking for a set of interests would be intrusive, imposing an extra work burden on the researchers, and thus result in a reduced uptake of the system. For this reason unobtrusive monitoring of web browsing via a proxy server was chosen to acquire positive examples of user interest. Optional feedback is elicited from both normal search results and recommendation lists, recording any information volunteered on the interest and quality of particular papers. This approach offers feedback options as an addition to the normal working of the system, and hence should not prove too intrusive. Profile visualization is available as an option to those who choose to use it, and is based on a drawing metaphor so that users can both see their current profile and literally draw any modification to it. Again, profile feedback is volunteered by users so should not interfere with their normal use of the system.

Research papers are represented as term-frequency vectors, a common representation in the machine-learning community. This allows use of a boosted k-nearest neighbour document classifier, described later in this chapter, to classify new vectors in terms of the classes within the topic ontology. Term frequency vectors are computed by counting the number of times words, or terms, appear within the textual content of a paper.

Recommender systems typically use a binary approach to classification, holding example of positive and negative interest for each user and classifying new papers based on how well they match these two training sets. One problem with this approach is that the training sets are personal to each user, so there is no easy basis on which they can be shared. Without the option of sharing, the number of training examples per user will be limited. This means that to gather a sufficiently large training set to allow accurate classification of interests, the users will either have to volunteer numerous examples of interest or be monitored for a significant duration to gather enough behaviour data. This places a significant burden on users to train the system, which will deter user uptake of the tool.

Foxtrot uses an ontological approach to interest profiling, representing user interests in terms of a research topic ontology. Since all users share the topic ontology, training examples of topic classes within the ontology can also be shared. A multi-class classifier is used to classify research papers in terms of the classes within this research paper topic ontology. Having represented interests in ontological terms we can then

use the relationships between classes to infer more interests than are available from direct observation only. Interest profiles can also be visualized using ontological terms, which are understandable to the users, and hence it is possible to elicit feedback on the profiles directly.

The interests held within each user's profile are computed from the research topics of browsed papers and from any feedback provided. A topic interest history is created for each user, and a time-decay function used to compute the current topics of interest. Inference is also applied, using the is-a relationships between classes defined by the research topic ontology to infer interest in more general classes when interest is seen in a specific class.

6.5.1 Research paper representation

Research papers are represented as term vectors, with term frequency / total number of terms used for a terms weight. Since many words are either too common or too rare to have useful discriminating power to the classifier, a few dimensionality reduction techniques are used to reduce the number of dimensions of the term-frequency vectors. Porter stemming [porter80] is used to remove term suffixes and the SMART [smart74] stop list is used to remove very common words. Term frequencies below 2 are removed and for each topic class only the top 50 terms, ranked by document frequency, are added to the vector. Dimensionality reduction is common in information systems; [sebastiani02] provides a good discussion of the issues.

Foxtrot supports papers in HTML, PS, PDF formats and various compressed versions of these formats, all of which are converted to plain text and then used to create term vectors. This covers the majority of research paper formats available on the web, with other formats or corrupt formats ignored. Several heuristics are used to determine if the research papers are converted to text correctly and look like a typical research paper with terms such as 'abstract' and 'references'. In the Foxtrot trial, term-vectors for papers had, after dimensionality reduction, around 1000 dimensions.

6.5.2 Research paper topic ontology

Foxtrot uses a research paper topic ontology to represent research interests of its users. A class is defined for each research topic and is-a relationships defined where appropriate. The ontology is based on the classification ontology used by the CORA

[mccallum00] digital library with a few customised modifications. Figure 6.2 shows some of the classes within the ontology. CORA was chosen, as opposed to the [dmoz] classification used by Quickstep, since it provided a readily accessible source of pre-classified research papers over a wide range of computer science topics, helpful when you need over 700 example papers.

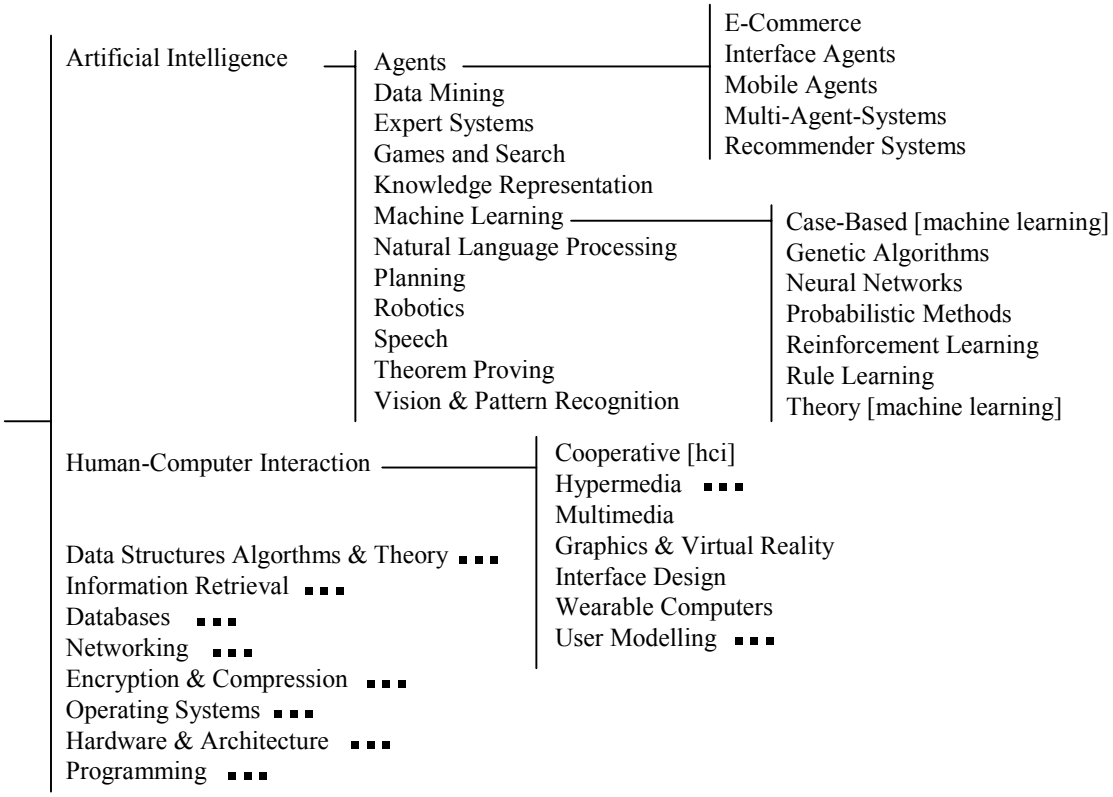


Figure 6.2 : Section from the research paper topic ontology

Approximately 5-10 labelled examples of research papers were manually added to the classifier training set, many taken from the results of the Quickstep trial and papers downloaded from the CORA system. There totalled 97 classes and 714 training examples. The ontology remains fixed through the Foxtrot trial, but could be updated as time goes on to reflect changes in the research domain. For every new ontology class a new set of 5-10 example papers would be required. Since the vector space used by the classifier is re-built every day, adding new examples mid-trial would not cause a problem to the system.

The effect changing ontology has on a system, i.e. from dmoz in Quickstep to CORA in Foxtrot, is a like the effect changing a document corpus has on a machine learning algorithm. The classification performance figures will be very much dependent on the

discriminating power of the training examples associated with each ontological class; this will in turn effect the profiling accuracy. Since the domains addressed by the CORA and dmoz ontology are similar, with CORA just being a little wider and broader in scope, we think the results from Quickstep and Foxtrot should be comparable. Certainly if the fit between ontological classes and the actual topics being read by users is good then the profiling accuracy will be good too, and thus the recommendation accuracy will be superior to a system where the ontological classes do not fit so well to the browsed topics.

6.5.3 Research paper classification

Research papers in the central database are classified by an IBk [aha91] classifier, which is boosted by the AdaBoostM1 [freund96] algorithm. The IBk classifier is a k-Nearest Neighbour type classifier that uses example documents, called a training set, added to a term-vector space. Figure 6.3 shows the basic k-Nearest Neighbour algorithm. The closeness of an unclassified vector to its neighbours within the term-vector space determines its classification.

$d_1 \ldots d_k$ are the k nearest documents to d_{new}

$$f(d_{new}) = \sum_{i=1}^{k} \frac{1}{w(d_{new}, d_i)}$$

$$w(d_a, d_b) = \sqrt{\sum_{j=1}^{T} (t_{ja} - t_{jb})^2}$$

$w(d_a,d_b)$	kNN distance between document a and b
d_a,d_b	document vectors
T	number of terms in document set
t_{ja}	weight of term j document a
f(d)	k-NN function

Figure 6.3 : k-Nearest Neighbour algorithm

Classifiers like k-Nearest Neighbour allow more training examples to be added to their term-vector space without the need to re-build the entire classifier. They also degrade well, so even when incorrect the class returned is normally in the right "neighbourhood" and so at least partially relevant. This makes k-Nearest Neighbour a robust choice of algorithm for research paper classification.

Boosting works by repeatedly running a weak learning algorithm on various distributions of the training set, and then combining the classifiers produced by the weak learner into a single composite classifier. The "weak" learning algorithm here is the IBk classifier. Figure 6.4 shows the AdaBoostM1 algorithm.

Initialise all values of D to 1/N
Do for t=1..T
call weak-learn(D_t)
calculate error e_t
calculate $\beta_t = e_t/(1-e_t)$
calculate D_{t+1}

$$\text{classifier} = \underset{c \in C}{\text{argmax}} \sum_{\substack{t = \text{all iterations} \\ \text{with result class c}}} \log \frac{1}{\beta_t}$$

D_t	class weight distribution on iteration t
N	number of classes
T	number of iterations
weak-learn(D_t)	weak learner with distribution D_t
e_t	weak_learn error on iteration t
β_t	error adjustment value on iteration t
classifier	final boosted classifier
C	all classes

Figure 6.4 : AdaBoostM1 boosting algorithm

AdaBoostM1 has been shown to improve [freund96] the performance of weak learner algorithms, particularly for the stronger learning algorithms like k-Nearest Neighbour. It is thus a sensible choice to boost our IBk classifier.

Other types of classifier were considered, including the naïve Bayes classifier and the C4.5 decision tree, and informal tests run to evaluate their performance. The boosted IBk classifier was found to give superior performance for this domain.

6.5.4 Interface

Users interact with Foxtrot via a web page. The basic interface is shown in figure 6.5. A web search engine metaphor has been used for the interface design, allowing users to enter search queries via edit boxes and click on a search button to initiate a search. Search results are returned in the area below the edit boxes, showing the details of each research paper found. Two sets of radio buttons appear below each search result to allow users to provide relevance and quality feedback if they so desire. When users first go to the Foxtrot web page, their daily recommendations are automatically

presented in the search result area. In this way, users can then choose to read the recommendations or just enter a search query and use the system normally.

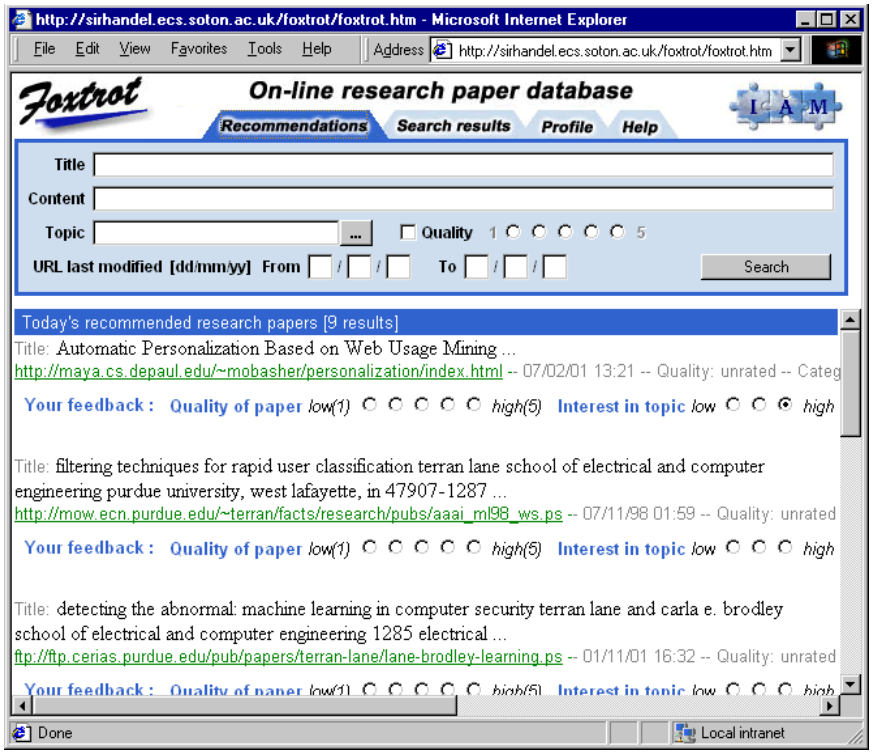


Figure 6.5 : Recommendation and search interface

Users who are in the profile group can visualize their interest profiles by clicking on a profile tab. Figures 6.6 and 6.7 show the profile interface. Profiles are displayed as a time/interest graph, showing what the system thinks their top interests are over the period of the trial. Direct profile feedback can be drawn onto this graph by using the controls to the side. A drawing package metaphor is used here, and users can draw coloured horizontal bars to represent a level of interest in a topic over a period of time. In this way a user can literally draw their own profiles.

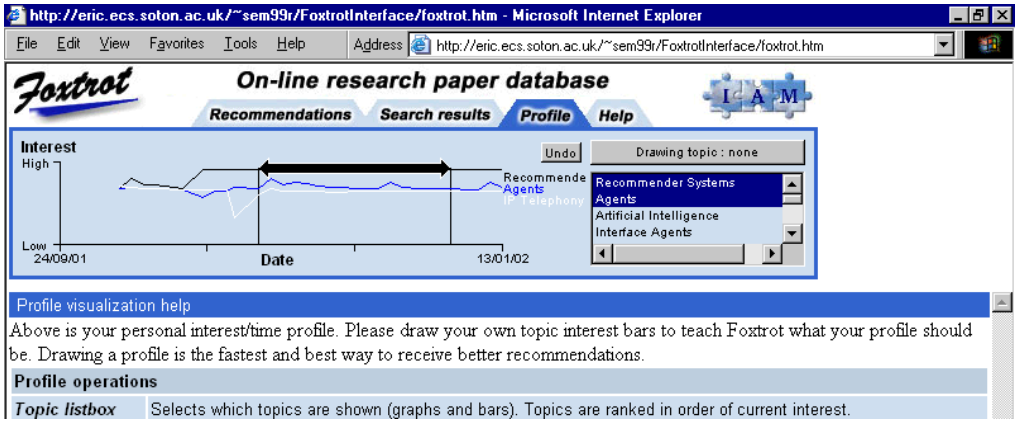


Figure 6.6 : Profile visualization interface, drawing interests

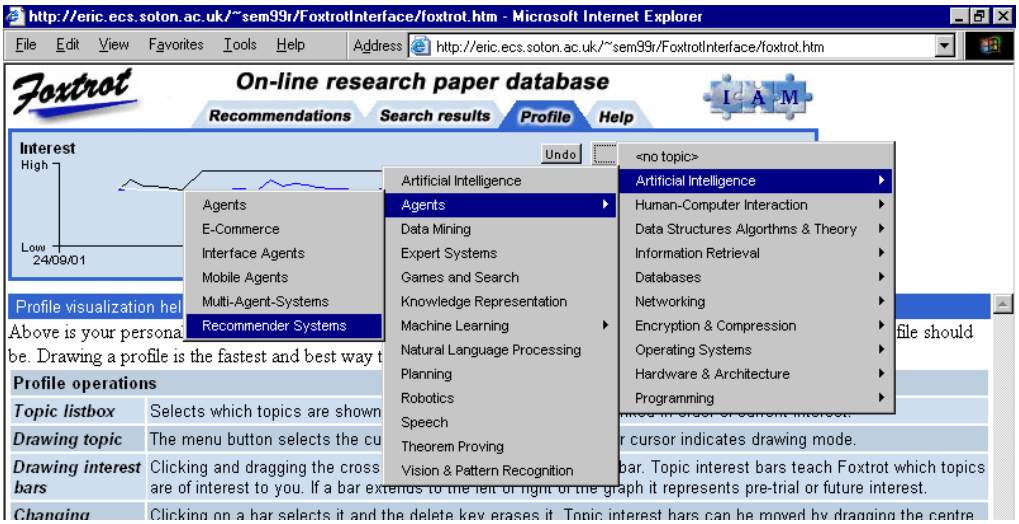


Figure 6.7 : Profile visualization interface, picking topics

In addition to the Foxtrot web page, a weekly email notification feature was added three months from the end of the trial. This provided a weekly email stating the top three recommendations from the current nine recommendations. Users could then jump to these papers or load the Foxtrot web page and review all nine recommendations. Figure 6.8 shows the email notification message.

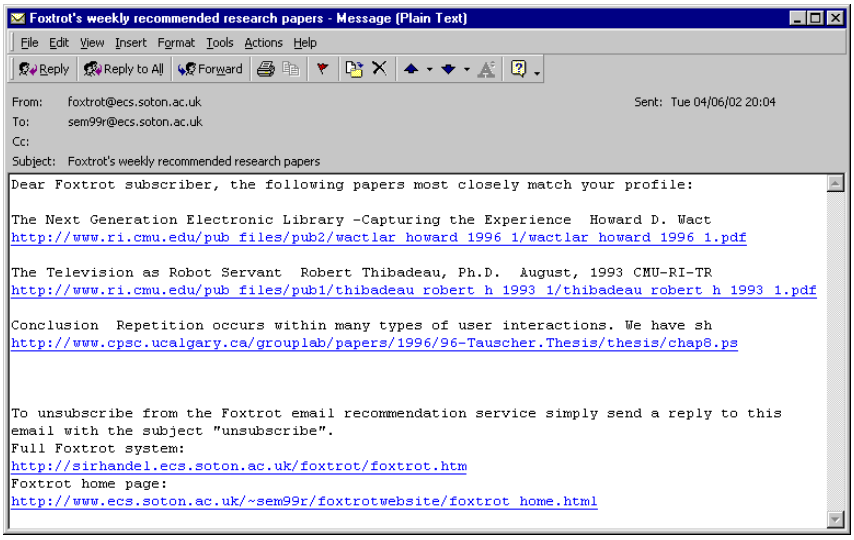


Figure 6.8 : Email notification interface

6.5.5 Profiling

Interest profiles are computed daily by correlating previously browsed research papers with their classification. User profiles thus hold a set of topics and interest values in their topics for each day of the trial. User feedback also adjusts the interest of topics within the profile and a time decay function weights recently seen papers as being more important than older ones. Ontological relationships between topics of interest are used to infer other topics of interest, which might not have been browsed explicitly; an instance of an interest value for a specific class adds 50% of its value to the super-class. Figure 6.9 shows the profiling algorithm.

$$\text{Topic interest} = \sum_{n=1}^{\text{no of instances}} \text{Interest value}(n) \,/\, \text{days old}(n)$$

Event interest values	Paper browsed = 1 Recommendation followed = 2 Topic rated interesting = 10 Topic rated not interesting = -10
Interest value for super-class per instance	= 50% of sub-class

Figure 6.9 : Profiling algorithm

6.5.6 Recommendation

Daily recommendations are formulated by a hybrid recommendation approach. A list of similar people to a specific user is compiled, using a Pearson-r correlation on the

content-based user profiles. Recommendations for a user are then taken from those papers on the current topics of interest, which have also been read by similar people to that user. Figure 6.10 shows the recommendation algorithm. During the Foxtrot trial three papers were recommended each day on the three most interesting topics, making a total of nine recommended papers. Previously read papers were not recommended twice and if more than three papers were available for a topic they were ranked by quality rating.

$$\text{Pearson r coefficient}_{ab} = \frac{\sum_{t=1}^{\text{topics}} (I_a(t) - \bar{I}_a) * (I_b(t) - \bar{I}_b)}{\sqrt{\sum_{t=1}^{\text{topics}} (I_a(t) - \bar{I}_a)^2 * \sum_{t=1}^{\text{topics}} (I_b(t) - \bar{I}_b)^2}}$$

$I_a(t)$ = User a's interest in topic t
$\bar{I}_a$ = User a's mean interest value over all topics

$\text{Pearson r coefficient}_{ab}$ = similarity of user a's profile to user b's profile
Recommended papers = papers on user's current interests $\cap$ papers read by similar users

3 papers recommended on the 3 most interesting topics, totalling 9 papers per day
If more than 3 papers meet above criteria, papers ranked by quality rating

Figure 6.10 : Recommendation algorithm

6.6 Experimental Evaluation

An experiment was carried out over the period of a single academic year to assess the performance of the Foxtrot recommender system. Test subjects were taken from both the staff and students of the computer science department at the University of Southampton. Foxtrot was used by the test subjects to assist in their everyday research. The overall recommendation and user profiling performance was measured, in addition to measuring the relative performance of those who used the profile visualization option.

6.6.1 Details of the trial

The user trial took place over the academic year 2001/2002, starting in November and ending in July. Of the 260 subjects registered to use the system, 103 logged onto the system, and of these 37 subjects used the system three or more times. All 260 subjects used the web proxy and hence their browsing was recorded and daily profiles built.

58 subjects joined the trial as it progressed, hearing about the system from advertising posters and word of mouth. At the start of the trial a bootstrap of about 6,000 documents where loaded into the central database by a web crawler. By the end of the trial the database had grown to 15,792 documents as a result of subject web browsing.

When registering to use the system subjects were randomly divided into two groups. The first 'profile feedback' group had full access to the system and its profile visualization and profile feedback options; the second 'relevance feedback' group were denied access to the profile interface. The objective of this subject grouping was to measure what difference, if any, visualizing profiles and providing profile feedback makes to the performance of the recommender system. It was found that many in the 'profile feedback' group did not provide any profile feedback at all, so in the later analysis these subjects are moved into the 'relevance feedback' group.

A binary comparison style experiment was chosen over other types of experimental design in order to test the relative effectiveness of the using profile feedback. It was felt that such a simple experimental design would yield the greatest chance of success, important for such a lengthy experiment with no chance of a re-run. Both groups used identical systems, except for one group's profiling features being disabled, in order to minimise effect causations other than the use of profile feedback. The university's major operating systems and web browsers were supported, along with a consistent web interface to allow subjects to use the system in their normal environment.

Towards the end of the trial, an additional email feature was added to the recommender system. This email feature send out weekly emails to all users who had used the system at least once, detailing the top three papers in their current recommendation list. Users could then follow URL links to the papers held in the email or go to the Foxtrot web page to see the full recommendation list of nine papers. Email notification was started in May and ran for the remaining three months of the trial.

Throughout the trial various metrics were logged. Jumps to paper URLs and relevance feedback were recorded in addition to profile feedback. Browsed URLs were also recorded via the web proxy. Relevance feedback on individual papers was recorded on a 5-point scale, as was quality feedback. Profile feedback generated a set of

interest bars, declaring a degree of interest in a topic over a period of time; it was possible for interests to be declared as ongoing. All recommended URLs and recommendations sent via email were recorded.

A post-trial questionnaire was send via email to all users who used the system at least once. A 5-point scale was used to record answers to the questions in the questionnaire.

6.6.2 Experimental Data

The raw data obtained from the trial occurs at irregular time intervals, based on when subjects looked at recommendations or browsed the web. For ease of analysis data is collated into weekly figures by summing interactions throughout each week. Group data is computed by summing the weekly contribution of each subject within a group. Figure 6.11 shows the metrics measured during the trial and derived values computed from them.

Explicit feedback
Interest = User declaring an interest on a paper URL
Jump = User following links to a paper URL
Profile feedback = User drawing a profile

Web proxy monitoring
Browsed URL = URL browsed by user

System logging
Recommended = URL recommended to user via web page
Email = URL recommended to user via email

Metric types
Interest = (URL,<interest type>,time)
Jump = (URL,time)
Profile feedback = (topic,<interest value>,start date,end date)
Browsed URL = (URL,time)
Browsed paper = browsed URL appearing in paper database
Recommended paper = (URL,time)
Emailed paper = (URL,time)
<interest type> = (high,average,low,unknown)
<interest value> = value between –10 and 10

Derived metric types
<future papers> = browsed/jumped papers in the 4 weeks after profile
<papers> = browsed/jumped papers over duration of profile (normally 1 day)
<top topics> = top 3 topics of profile

$$\text{Predicted profile accuracy} = \frac{\text{No of <future papers> matching <top topics>}}{\text{No of <future papers>}}$$

$$\text{Profile accuracy} = \frac{\text{No of <papers> matching <top topics>}}{\text{No of <papers>}}$$

$$\text{Web page rec accuracy} = \frac{\text{No of recommended papers browsed or jumped to}}{\text{No of recommended papers}}$$

$$\text{Email rec accuracy} = \frac{\text{No of emailed papers browsed or jumped to}}{\text{No of emailed papers}}$$

$$\text{Jumps to recommendations ratio} = \frac{\text{No of jumps to recommended papers}}{\text{No of jumps}}$$

$$\text{Jumps to profile topics ratio} = \frac{\text{No of jumps to papers matching <top topics>}}{\text{No of jumps}}$$

$$\text{High interest ratio} = \frac{\text{No of high interest feedback cases}}{\text{No of interest feedback cases}}$$

$$\text{Low interest ratio} = \frac{\text{No of low interest feedback cases}}{\text{No of interest feedback cases}}$$

Figure 6.11 : Metrics measured during the Foxtrot trial

Subject selection for the ‘profile feedback’ group was taken from those subjects who had provided profile feedback, with all other subjects placed into the ‘relevance feedback’ group. The subject selection could be made more favourable by only including subjects who used the system a few times, but it was felt that including

subjects who gave up on the system would be more representative of how the trial went.

The total number of subjects who registered with the system was 260, of which nine were in the 'profile feedback' group and 251 were in the 'relevance feedback' group. Of these 260 subjects 37 used the system three or more times, making a system uptake rate of 14%. This is representative of the fact that the system was offered as an optional tool that could be used in addition to normal work practices; as such only a fraction of potential subjects chose to invest the time and effort required to get the most out of the system.

Basic trial measurements come from the recorded subject interaction with the recommender web page and logged data from the web proxy.

Explicit feedback includes jumping to search results and jumping to recommendations via the web page interface, interest feedback on search results and feedback on profiles via the profile visualization interface.

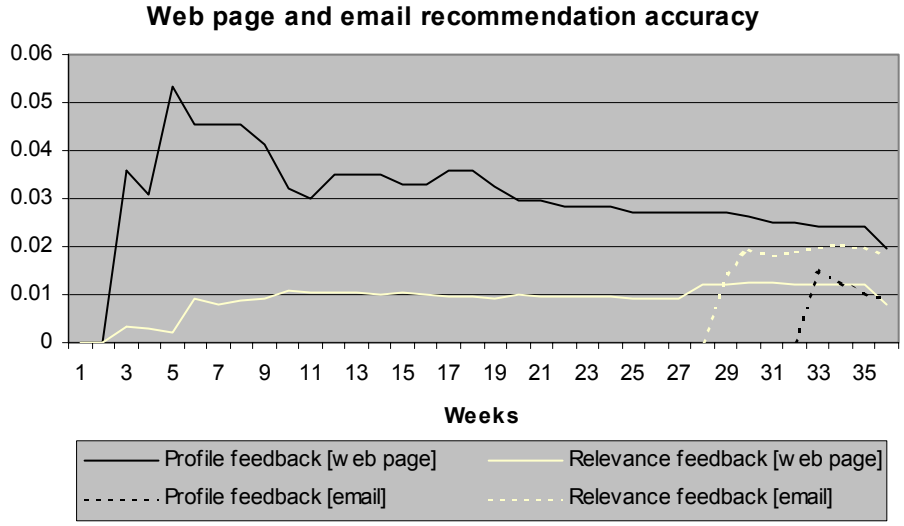
Implicit feedback is obtained from unobtrusive monitoring via the web proxy. The proxy logs are parsed to extract the subject's browsed URLs, which are recorded with a timestamp. These URLs are later correlated with the research paper database to obtain a set of browsed research papers.

In addition to feedback, the system records when web page and email recommendations are issued. This allows various derived metrics to be computed to compare the relative effectiveness of different types of system operation.

The recommendation accuracy metric takes explicit feedback and computes accuracy figures for both web page and email recommendations. A simple ratio is used to obtain the number of recommendations followed as a fraction of the total number of recommendations; this provides a measure of the effectiveness of the recommendations. Figure 6.12 shows the recommendation accuracy for web page and email recommendations.

The small number of subjects within the 'profile feedback' group accounts for the larger confidence intervals. While not statistically significant, there is an apparent

trend for more accurate recommendation when using profile feedback, especially in the earlier weeks. Email recommendations appeared to be preferred by the 'relevance feedback' group, slightly out performing the 'profile feedback' group.

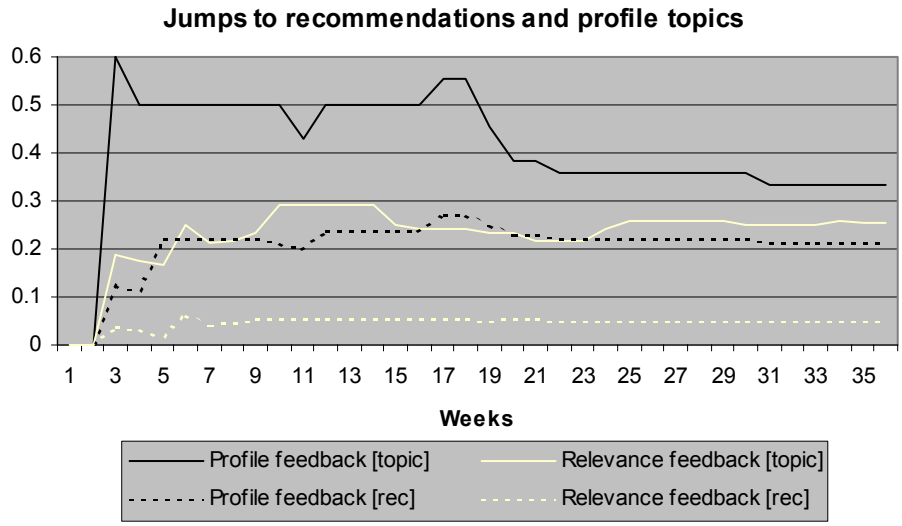


Final week 95% confidence intervals			
Profile feedback [web page recs]	0.043	Relevance feedback [web page recs]	0.0028
Profile feedback [email recs]	0.018	Relevance feedback [email recs]	0.011

Figure 6.12 : Web page and email recommendation accuracy

In addition to recommendation accuracy, the ratio of jumps to recommendations against all jumps total and the ratio of jumps to papers with a top three topic against all jumps were computed. Jumps to recommendations measure the degree to which subjects use the recommendation facility as opposed to just using the search facility of the database. Jumps to papers with a top three profile topic measures how well the profiles fitted the subject's actual interests. Figure 6.13 shows the figures for jumps to recommended papers.

The 'profile feedback' group made a greater proportion of jumps to recommendations than the 'relevance feedback' group; this trend is statistically significant. A similar trend is seen in jumps to papers on top profile topics, but is less clear.

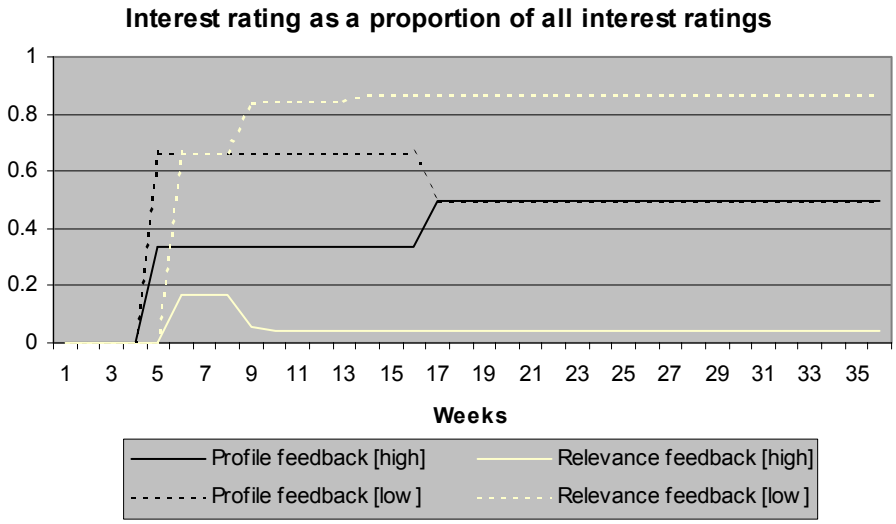


Final week 95% confidence intervals			
Profile feedback [topic]	0.27	Relevance feedback [topic]	0.041
Profile feedback [rec]	0.14	Relevance feedback [rec]	0.039

Figure 6.13 : Jumps to rec's and profile topics as a ratio of all jumps

Interest feedback is primarily used by the system to compute profiles, but it does also provide an indication as to the utility of recommended papers. If subjects rate pages as highly interesting then this is a sign the recommendations are good. Figure 6.14 shows both high and low interest ratings as a proportion of all interest ratings.

Subjects were reluctant to provide feedback, with just 72 interest ratings being recorded during the trial. Only two 'profile feedback' subjects provided interest feedback, hence the larger confidence interval. Even so, the 'profile feedback' subjects clearly expressed a higher level of interest in their papers than the 'relevance feedback' group, who rated most papers as of low interest. In both groups there was a trend for subjects to provide all their interest feedback in the first few weeks after registering with the system.

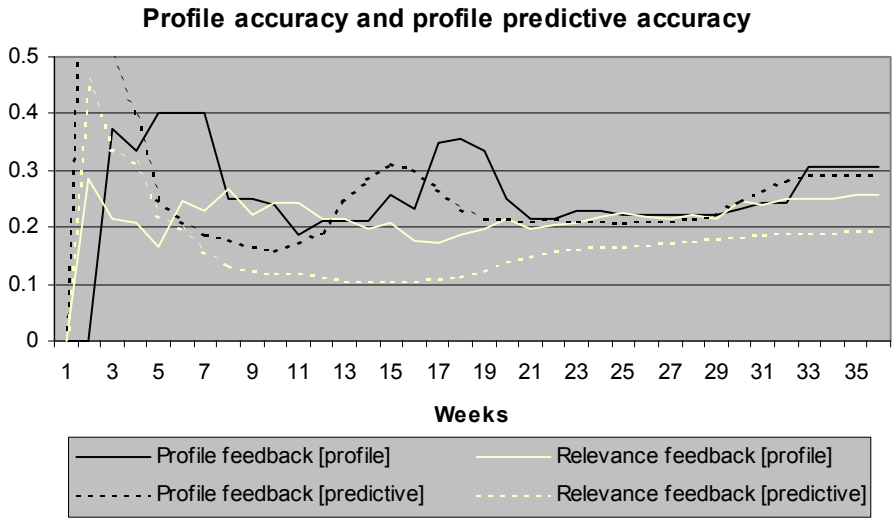


Final week 95% confidence intervals			
Profile feedback [high]	0.3	Relevance feedback [high]	0.011
Profile feedback [low]	0.3	Relevance feedback [low]	0.017

Figure 6.14 : Interest ratings as a ratio of all interest ratings

Since user browsing is recorded, both the profile accuracy and profile predictive accuracy can be measured. Profile accuracy measures the number of papers browsed or jumped to that match the top three profile topics for the duration of that profile; since profiles are normally updated daily, the average life of a profile is one day. This is a good measure of the accuracy of the current interests within a profile at any given time. Profile predictive accuracy measures the number of papers browsed or jumped to that match the top three profile topics in a four-week period after the profile was created. This measures the ability of a profile to predict subject interests. Metrics are measured for every profile computed over the period of the trial, providing a view on how the quality of the profiles varies over the length of the trial. Figure 6.15 shows the figures for the profile metrics.

While not statistically significant, there is a trend for the 'profile feedback' group to have profiles that are better at predicting future browsing interests. This trend is not reflected in the daily profile accuracy figures however, where the two groups are similar. This would appear to show that the two groups are profiling slightly different interest sets, with the 'profile feedback' interests of a longer term nature.



Final week 95% confidence intervals			
Profile feedback [profile]	0.23	Relevance feedback [profile]	0.036
Profile feedback [predictive]	0.23	Relevance feedback [predictive]	0.036

Figure 6.15 : Profile accuracy and profile predictive accuracy

In addition to measuring subject group interactions with the system, the AdaBoostM1 boosted IBk classifier performance was computed. A standard cross-validation test was applied to the classifier training set, to obtain the figures for precision and recall. Table 6.1 shows the results. The precision value is a measure of how many correctly classified documents there were as a proportion of the number classified. The recall value is a measure of how many documents were classified as a proportion of the total number of documents.

	Precision	Recall	Number of classes	Number of examples	Number of terms
Classifier	0.42	1.0	97	714	1152

Table 6.1 : Foxtrot classifier precision and recall

6.6.3 Post-trial Questionnaire

A post trial questionnaire was sent out via email to every subject who used the system at least once. Table 6.2 shows the results of this survey, completed by 13 subjects. It clearly shows that the search facility was most useful to the subjects, with the recommendation facility being only partially used. This is borne out by the relatively small amount of feedback provided by users during the trial.

Question	1	2	3	4	5	Mean
How useful did you find the Foxtrot database?	4	2	5	2		2.38
How much did you use the recommendation facility?	7	5		1		1.62
How accurate were the recommended topics?	3	3	2	3	1	2.67
How useful were the recommended papers?	4	2	4		2	2.5

Table 6.2 : Foxtrot post trial questionnaire results

6.6.4 Discussion of the trends seen in experimental data

Generally speaking the 'profile feedback' group outperformed the 'relevance feedback' group for most of the metrics. The experimental data reveals several specific trends however.

Web page recommendations, and jumps to those recommendations, were better for the 'profile feedback' group, especially early on in the first few weeks after registering. This is probably because the 'profile feedback' users tended to draw their interest profiles when they first registered with the system, and only update them occasionally. This has the effect that the profiles are most accurate early on and become dated as time goes on. This aging effect on the profile accuracy is shown by the 'profile feedback' group performance gradually falling towards that of the 'relevance feedback' group as time goes on.

One interesting observation is that the initial performance enhancement gained using profile feedback appears to help overcome the cold-start problem [middleton02], a problem inherent to all recommender systems.

Email recommendation appeared to be preferred by the 'relevance feedback' group, and especially by those users who did not regularly check their web page recommendations. A reason for this could be that since the 'profile feedback' group used the web page recommendations more, they needed to use the email recommendations less. There is certainly a limit to how many recommendations any user needs over a given time period; in our case nobody regularly checked for recommendations more than once a week.

The overall recommendation accuracy was between 1-2%, 2-3% for the profile feedback group. This may appear low, especially when compared to other recommendation systems such as Quickstep, but it reflects the optional nature of the

recommendation service offered. Users had the choice to simply ignore recommendations if they did not help to achieve their current work goal. This optional nature of the system assisted system uptake and acceptance on a wide scale. In comments made via the post-trial questionnaire some users preferred email recommendations because they could address emails when they were not very busy, as opposed to when they were searching for specific papers for a particular task.

The profile accuracy of both groups was similar, but there was a significant difference between the accuracy of profile predictions. This probably reflects the different types of interests held in the profiles of the two groups. The 'profile feedback' group's profiles appeared to be longer term, based on knowledge of the user's general research interests provided via the profile interface. The 'relevance feedback' profiles were based solely on the browsing behaviour of the user's current task, hence contained shorter-term interests. Perhaps a combination of profile feedback-based longer-term profiles and behaviour-based short-term profiles would be most successful.

The overall profile accuracy was around 30%, reflecting perhaps the difficulty of predicting user interests in a real multi-task environment. Perhaps integrating some knowledge of which task the user is performing would allow access to some of the other 70% of their research interests. These interests were in the profile but did not make it to the top three topics of current interest.

The post-trial questionnaires showed that the overall usage of the recommendation facility was limited, with users preferring to use just the search facility. This was also born out by the interest rating results, where most 'relevance feedback' users appeared unhappy with their recommendations. Profile feedback users appeared to be more content with their recommendations, and tended to check recommendations regularly for about a week or two after drawing a profile. This appeared to be because users had acquired a conceptual model of how the system worked, and wanted to keep checking to see if it had done what they expected. If a profile was required to be drawn before registering on the system, this behaviour pattern could be exploited to increase system uptake and gain some early feedback. This may in turn increase initial profile accuracy and would certainly leave users with a better understanding of how they

system worked, beneficial for both gaining user trust and encouraging effective use of the system.

6.7 Comparison with other work within the literature

Group Lens [konstan97] is an example of a collaborative filter, recommending newsgroup articles based on a Pearson-r correlation of other users' ratings. Fab [balabanović97] is a content-based recommender, recommending web pages based on a nearest-neighbour algorithm working with each individual user's set of positive examples. Foxtrot is a hybrid recommender system, combining both these types of approach.

Personal web-based agents such as NewsDude [billsus98] and NewsWeeder [lang95] build profiles from observed user behaviour. These systems filter new stories and recommend unseen ones based on content. Personal sets of positive and negative example are maintained for each user's profile. In contrast, by using an ontology to represent user profiles Foxtrot pools these limited training examples for all its classes.

Digital libraries classify and store research papers, such as CiteSeer [bollacker98]. While Foxtrot is a digital library, its content is dynamically and autonomously updated from the browsing behaviour of its users.

Very few systems in the recommender system literature perform user trials using real users, making direct comparison difficult. Most use either labelled benchmark document collections to test classifier accuracy or logged user data taken from sources such as newsgroups. NewsWeeder reports a 40-60% classification accuracy with real users, while Personal Webwatcher [mladenić96] reports a 60-90% classification accuracy using benchmark data. Foxtrot's classifier is on the low side with 41% accuracy, but this appears much better when the number of classes used in classification is taken into account and the potential this allows for improving profiling via inference and profile feedback.

The Quickstep [middleton01b] system had a recommendation accuracy of about 10% with real users, and provides a useful system for comparison. Foxtrot manages 2-3% recommendation accuracy, which reflects the different types of subjects involved in the two trials. The Quickstep subjects were willing researchers taken from a computer

science laboratory, while the Foxtrot subjects were staff and students of a large department who would only be willing to use the system if it was perceived to offer direct benefits to their work. A recommendation accuracy of 3% means that roughly one in three sets of recommendations contained a paper that was downloaded. While initially appearing low, this result is good when the problem domain is taken into account; most systems in the literature do not attempt such hard and realistic problems.

Mladenic [mladenić99] provides a good survey of text-learning and agent systems, including content-based and collaborative approaches. Chapter 2 and appendix A lists many recommender systems in the literature today. The systems most related to Foxtrot are Entrée [burke00], which uses a knowledge base and case-based reasoning to recommend restaurant data, and RAAP [delgado98] that uses categories to represent user profiles with unshared individual training sets for each user. None of the reviewed systems use ontological categories to represent user profiles.

6.8 Conclusions from the Foxtrot trial

The experiment detailed in this chapter provides empirical evidence as to the effectiveness of using an ontological approach to user profiling in recommender systems. As with the predecessor system Quickstep, Foxtrot uses an ontology to represent user profiles, allowing training examples to be shared and knowledge of interests inferred without the need for direct observation.

Profile visualization and profile feedback were explored as mechanisms to further improve the profiling process, and were found to enhance both profiling accuracy and the resulting recommendation usefulness. The ontological approach to profiling provides a suitable basis to create a profile visualization that is understandable to users.

The overall performance of the Foxtrot system was found to be favourably comparable with other recommender systems when the difficult real-world problem domain was taken into account. While the recommendations were far from perfect, the system did provide a useful service to those users who chose to invest time and effort in using the system.

Individual aspects of the system could be enhanced further to gain a relatively small performance increase, such as increasing the training set size, fine tuning the ontological relationships and trying alternative classification algorithms. However, the main problem is that the systems profiler is not capturing about 70% of the user's interests. Obtaining an understanding of the current task a user is performing would be advantageous here.

Knowledge of a user's current task would allow the profiler to distinguish between short and long term tasks, separate concurrently running tasks and adjust recommendations accordingly. While 70% of users' browsing interests were not in the current profile's top three topics, they were in the profile somewhere at a lower level of relevance. Having separate profiles for each user task would allow a finer grained profiling approach, significantly improving performance.

It is interesting to note that the visualization of profiles could be extended to visualize the systems current task for a user. In this way direct task feedback by users could be elicited in much the same way as profile feedback can be. Even so, task modelling is far from easy to achieve in practice, but it does appears to be an important aspect of user profiling and one that future versions of this system may well investigate.

The Foxtrot recommender system described in this chapter uses an ontological approach to profiling, allowing profiles to be visualized and profile feedback acquired. It was shown that visualizing profiles allows performance gains in profiling and recommendation accuracy. Since profile visualization is a direct benefit of taking an ontological approach to user profiling, this chapter thus provides the evidence to prove the third sub-hypothesis of this thesis.

Chapter 7 Conclusions and future work

Chapter summary
Evidence from work in this thesis is collated in light of each sub-hypothesis.
The central hypothesis is proved.
The future direction of this work is discussed.

Information overload on the World Wide Web is an ever-growing problem. There are too many pages to browse effectively, and searching requires an explicit search query, which is all too often difficult to construct. What is needed is knowledge of where to look for information and what information is worth looking at.

Recommender systems learn the type of thing that you and others are interested in. They then use the knowledge from other people to recommend new items of information that you have not seen before, which similar people to you have found helpful before. Being collaborative in nature the knowledge of where to look and what is good is pooled for the common good.

In conjunction with a traditional search service, recommendation is a powerful tool that can increase the effectiveness of everyday work, removing some of the burden involved with finding new information and regularly checking existing information.

7.1 Conclusions

The **central hypothesis** of this thesis is that representing user profiles using an ontology offers advantages over traditional profile representations. In order to prove this hypothesis, this thesis describes three sets of experiments aimed at proving three sub-hypotheses.

The **first sub-hypothesis** is that inferences gained about profiles outweigh the loss of information that occurs when a profile is constrained to use only ontological terms.

Chapter 4 described the Quickstep recommender system and two trials that investigated what difference using an ontological approach to user profiling made when compared to a more basic flat-list of topics. Profiles represented using the ontology were able to infer new interests, thus adding interests to the profile that

could not be observed directly. The ontology also provided a mechanism for users to share training examples and gave a common structure to each user's interpretation of the classification of research papers.

Experimentation found that not only did the ontology group exhibit superior performance, but the overall performance of the recommender system was comparable with recommender systems reported within the literature. Therefore, the loss of information that occurs by using ontological representation was outweighed by the profile accuracy gained from inference. In short, an ontological profile representation is at least as good as the other, more traditional representations in the recommender system literature.

The **second sub-hypothesis** is that using an ontology allows other knowledge-bases, which use the same ontological concepts, to be used to draw on additional information not normally available to the recommender system.

Chapter 5 describes an integration of the Quickstep recommender system with a knowledge-base made up from personnel and publication information. The knowledge-base is used to reduce the recommender system cold-start effect by bootstrapping new profiles to the system. Because the ontological concepts in the profiles appear within the knowledge-base, direct inference about potential topics of interest to a new user can be made.

Experimentation found that the Quickstep recommender system cold-start problem could be significantly reduced by bootstrapping from the knowledge-base. As such this proves the second sub-hypothesis, and in doing so shows that representing a profile using an ontological representation has not just comparable performance, but clear advantages over more traditional representations.

The **third sub-hypothesis** is that explicit feedback on user profiles offers advantages over traditional relevance feedback.

Chapter 6 describes the Foxtrot recommender system and an experiment that investigates whether visualizing a profile and acquiring profile feedback increases the performance of a recommender system. The user profiles could only be visualized because they were represented in ontological terms, terms that the users were able to

understand. The ontology again allowed for inference to discover interests that were not observed directly, but this time there was an additional mechanism to tie profiles to the interests stated in the profile feedback provided by users.

Experimentation found that those users that provided profile feedback received more useful recommendations and had better profiles than those users that relied only on relevance feedback. This proves the third sub-hypothesis, and shows an additional advantage that ontological profile representation can offer over more traditional profile representation methods.

Taking the evidence shown from the three sub-hypotheses into account, it is clear that the central hypothesis of this thesis is thus proved. Representing user profiles using an ontology does offer advantages over traditional profile representations.

7.2 Future direction of work

7.2.1 Incremental improvements

The way to make immediate improvements in the performance of either the Quickstep or Foxtrot recommender system is fairly clear.

Increasing the size of the training sets used will improve classification accuracy, as would adjusting some of the classifier algorithm parameters such as the number of boosting iterations or the value of k for the k-nearest neighbour algorithm. The classification algorithm itself could be changed too, but any of these alterations would probably only result in a small total increase of no more than 10% in the accuracy figures. The effect this would have on the profiling and recommendations would be less noticeable still.

The profiling algorithm could be improved, taking into account the shape of the time-interest behaviour graphs that are acquired for each profile topic. This would increase the profiling accuracy slightly, but the simple time-decay function is good at capturing current interests anyway. The other transient or task related interests, missed by both the Quickstep and Foxtrot recommender systems, would still be missing from the profile.

The Pearson r recommendation algorithm is well used within the literature, and performs very well. As such it is hard to see much improvement being made to the basic collaborative recommendation algorithm. However, alternative algorithms could be tried.

All these incremental improvements could easily be made, but the overall performance gain would still be relatively small, leaving the recommender system still far from 100% profiling accuracy.

7.2.2 Fully exploiting the ontology

The existing Quickstep and Foxtrot recommender systems use only a simple is-a ontology. There is a potential for significant improvement in profiling accuracy by maintaining more relationships that model various aspects of the academic domain. Knowledge about concepts such as related research fields, common technologies within a field, key authors and their publications and many more could all be used to infer significantly more about a user's browsing behaviour than just the topic.

Both the "project" relationship, indicating who is a member of which project, and the "research area" relationships are perhaps the most promising in this respect. In an academic or commercial organization there are likely to be many projects at any given time, and the people on those projects are likely to share at least some project related interests. Projects often have well defined research areas, so this links in "research area" which already has strong links to academic paper classification. A rich, well-maintained ontology could also link in well with task knowledge discussed in the next section, maintaining knowledge about people, projects and the related technologies they use.

Of course, maintaining such a large ontology is always a difficult and expensive task. One potential for improving system scalability is to use modern clustering techniques to discover classes for the ontology automatically, maybe as a support tool or consistency checker. Certainly it would be possible to apply clustering to an existing ontology of research topics to find areas of research where new topics are emerging, flagging an area of the ontology as ripe for analysis and potentially ready for splitting into a sub-topic or new area completely. Tools such as this would make ontology

maintenance more automatic, reducing the end cost in terms of human effort; this would in turn improve the scalability of using ontologies in recommender systems.

7.2.3 Applying task knowledge to the user profiling

The next step to make a significant improvement in profile accuracy, and hence recommendation usefulness, is likely to be exploring the concept of task modelling. If the profiling algorithm can be made more task orientated, then the 70% or so of interests missed by the Foxtrot system could be recognised and assigned to tasks other than general interest. This would mean that the overall recommendations could be more focused and of more active help to users' everyday work, rather than something they just check to keep up-to-date like a general reading list. Evidence for the need to introduce task modelling can be found in chapter 7, from the results of evaluating the Foxtrot recommender system.

This is far from a trivial task as acquiring knowledge of what tasks the user is doing and when is particularly difficult to do in a non-obtrusive way. However, if recommender systems, and in particularly user profiling methods, are to make major progress in the future it is an area of research that must be addressed.

7.2.4 Utilizing the agent metaphor

A recommender system is really a type of interface agent [maes94]. All that would be required to make the Quickstep or Foxtrot recommender systems into an interface agent is to encapsulate the system in terms of an agent wrapper, with one copy of the agent made for each user. The user's recommender agent could then offer a set of recommendations to its user, share knowledge of its user's preferences with other trusted agents and participate in a multi-agent systems, perhaps to trade information about the user in order to build a better profile and acquire a better set of recommendations.

Such a multi-agent system could provide the framework to bring in ontological knowledge from other sources and task knowledge about the user. Potentially this could be a very powerful technology indeed.

Another aspect of the interface agent metaphor is the potential for indirect HCI [kay90]. Traditional direct manipulation style interfaces use a simple metaphor, such

as the desktop, to provide visual iconic representations of different aspects of the computer, such as a filing cabinet. Actions can be sent to the computer by directly manipulating icons via a mouse, and the results of actions seen immediately via direct feedback. Indirect HCI is the opposite, where the user tells a trusted agent what is required, for example "download all document files on the alpha project from this ftp site", and the agent carries out the users wishes behind the scenes. There are obvious trust issues here and the need for users to feel in control, but indirect HCI offers significant potential alongside direct manipulation as a power interface metaphor. In many respects, the recommendation process is an indirect HCI process.

7.2.5 Social implications

Recommendation is not a new thing, since all that is required is a social medium for which people can share ideas and value judgments. Internet discussion groups have served this purpose for some time, as has email communication and simple verbal exchanges in the company coffee room. A recommender system is really an automated system that avoids some of the time consuming activities associated with traditional media, such a regularly checking a bulletin board. As such, the implementation of a recommendation service to an organization could have an impact on the traditional social fabric within that organization. Would people feel less need to participate in normal social activity, or would the recommendation facility simply serve as an additional source of value judgements? Can recommendations be trusted on sensitive issues, and who would compensate those people who acted on bad information? These sorts of social issues have not been investigated in the literature today, but could have a significant impact to an organization proposing to use such a system.

7.3 The nature of ontology and recommender systems

Recommender systems and ontologies hold, as we explored in chapter 5, many complementary features. Recommender systems are driven by the need to model dynamic, behaviour information about people and are well placed to respond quickly to the inevitable changing nature of people's work. In many ways a recommender system provides a window into the current state of a social organisation.

Ontologies on the other hand are mostly driven by knowledge engineering, representing statically true facts about a domain. The knowledge they represent is specifically chosen not to become out of date too quickly, and therefore tends to be constantly useful over long periods. In this respect an ontology is a well engineered window on the historical and more permanent state of an organisation.

Integrating these two approaches offers the potential for significant step forwards in both areas. Recommender systems can overcome the cold start problem through ontological bootstrapping, and if closely linked to external ontological structures can respond dynamically to occasional, well-considered high-quality changes to those ontological structures. Such ontological changes will occur as the years go by and an organization changes, and provide a mechanism to keep the recommendation framework up-to-date. On the flip side a recommender system will be responding to the here and now, and as such is an ideal source of up-to-date information for an ontology. It could also be used to identify emerging areas of ontological interests, such as new research fields, through modern clustering techniques, for example, to identify potential new classes. This could be fed back to the maintainers of the ontology to trigger investigation and possible future change.

Recommender systems can thus drive ontological population, with dynamically acquired information, and ontological development, through the identification of emergent domain structure. Ontologies in return can bootstrap recommender systems and provide a slowly evolving, well-considered structure for its recommendation framework that closely matches the organisation it supports.

7.4 Recommendation is here to stay

Throughout history people have learnt from the experience of others, and with the information systems of today the situation is no different. Be it the sharing of web URLs, recommendation of exciting authors or simple gossip about the number one music CD, recommendation is a social process we all participate in. It really does not matter how much information is out there, be it on web pages, databases or rich well maintained ontologies, the most useful information of all is the knowledge from other experts as to what is worth knowing and what is not. This is why recommendation

plays a role in the information systems of today, and will continue to do so in the information systems of tomorrow.

Appendix A Summary of reviewed systems

What follows is a summary of the recommender systems reviewed in this thesis. The recommender systems are listed by application domain, so that similar types can be compared together. The machine-learning terminology used here is described in the glossary.

When reviewing commercial systems the exact algorithms used are often not published, so can only be surmised. A more detailed review of commercial E-commerce recommender systems can be found in [schafer99].

Quantitative results are detailed where seen so that comparisons (however difficult) can be drawn between systems.

E-commerce domain

Amazon.com is a commercial book shop/recommendation service. Customers can rate books they have read using a five star rating, and attach a textual review of the book. This feedback information is shared and used to collaboratively recommend books to other users. Recommendations are based on either the most frequently purchased books or books purchased by similar people to the current user (based on a match between the current user's previously purchased books and other user's previously purchased books).

In addition to the recommendation service, a conventional search engine can be used to find specific books.

Dietorecs [arslan02] is a recommender system for travel products and services. Users fill out a set of requirements, which form the basis of a logic filter of potential options. Case-based reasoning is used to identify the closest case within a case-base. A similarity function is then used to determine a likely set of items that can be recommended.

eBay is a commercial system which allows buyers and sellers to contribute to profiles of other customers with whom they have done business. A satisfaction rating is elicited from users, which is shared and used when recommending potential sellers.

EFOL [svensson01] is a shopping program in which recipes are selected and the ingredients ordered on-line. Collections of recipes are created by users (using an editor) and made available to others. Discussion about individual choices of recipe is facilitated using a chat area. Collaborative recommendation is supported where clusters of similar users are formulated (using a system editor) and made available for others to take suggestions from.

Results: 12 people (all researchers) used the system on two separate occasions. Half the users reported other people's recipes influenced them, and the pictures of food made them feel hungry.

Entrée [burke00] is a recommender system for a restaurant database. It uses a hand-crafted knowledge-base of local restaurant information to perform similarity matching based on user queries. The use of a knowledge-base reduces the cold-start problem.

Results: three year trial, 20,000 system interactions, random 55-60% recommendation precision, collaborative filtering algorithm 65-70% precision.

Levis is a commercial clothing recommender system. Feedback on three categories (music, looks and fun) is elicited from the user using a 7-point scale. Six items of clothing are then recommended from the Levis range and feedback elicited on the recommendations. Recommendations are thus based on what similar people preferred.

FAIRWIS [buono01] recommends trade fare information via an on-line catalogue system. Interaction with the system such as web page access or printing is logged and user profiles created. A Pearson r correlation finds similar profiles. Groups of similar users are then stereotyped and the on-line information customized based on the assumptions made for that stereotype.

Ghani [ghani02] describes a recommender system for an on-line clothing retailer. A knowledge base of clothing information is maintained, recording the web pages of various clothing retailers with price, clothing etc. information. Using a training set of about 700 pages with log-odds ratio term weights, Expectation-Maximization is used to dynamically create a profile for the user bases on their web browsing. Real-time recommendations are thus generated to help the user, based on this dynamic profile.

LIBRA [mooney00] is a book recommender system. The Amazon.com database of books is taken as a labelled dataset and represented using a bag-of-words representation after removing stop-list words. Users provided book ratings on a 10 point scale and a naïve Bayes classifier used to classify books as interesting or not.

Results: 0.49 to 0.88 precision depending upon book genre in Amazon.com book dataset.

MIAU [bauer02] recommends items from an electronic catalogue on cars. System interactions and explicit feedback build up a value tree for catalogue items, forming the basis of a user model. Stereotypical user profiles are also added to the system to bootstrap it initially. A statistical average profile is computed from those users whose profiles do not conflict with the current users profile. This average profile is used as the basis for recommendations.

RIND [cöster02] is a recommender system to help with buying a PC. Hardware configurations that other people have bought are recorded into a database. Users provide attributes that they require from their computer, forming a feature vector. A k-nearest neighbour classifier and naïve Bayes classifier are then used to compute the most similar configuration option to the feature vector.

Ski-europe.com [delgado02] recommends ski holidays. User interaction with the ski web site is recorded to form a user profile. Both a short-term session profile and longer-term historical profile is formed. A cosine similarity algorithm is used to provide ratings on potential holiday options, and thus recommendation of specific packages.

Email filtering domain

Tapestry [goldberg92] is an email recommendation system. Users annotate documents as they read them, and collaboratively share this information with others. Users' email habits are monitored and implicit feedback obtained (such as user x replied to email y). Users can program filter rules into the system, which are regularly executed. Rules are things like "I want all documents read by user x". As the implicit feedback is shared, the had-crafted rules are simple collaborative filters. A SQL like language allows users to enter rules.

Expertise finder domain

Expertise Recommender [mcdonald00] assists technical support help desk staff in finding the right expert for a task. Recommendations are based on prior requests, so people who helped successfully with a problem before can be selected by the user to do so again. The user can choose which pre-programmed heuristics are used to recommend suitable people (e.g. select by minimum current workload).

Hand crafted heuristics mine an eight-year change-control database to extract initial profile records of who can handle what sort of problem. Profile records are stored with a vector-space model, with features identified using a set of thesauri (a stop list is used but no stemming applied).

The recommender only filters the set of possible people to handle a task. An interactive interface allows the help desk staff to pick the particular person they need.

Referral Web [kautz97] models a social network by monitoring social communication sources (email, net news, home pages etc.) and extracting a network model. Heuristics extract names of other people from an individual's communications, which are then refined by computing the Jaccard coefficient between the individual and other names. Once built, the social network can be browsed, and questions asked of it. Recommendations of related people to talk to about an area can thus be extracted (e.g. list documents close to Tom Mitchell).

Movie domain

CBCF [melville02] Content-boosted collaborative filtering (CBCF) is a technique used by Melville to produce a movie recommender system. A naïve bayes classifier is run on the EachMovie dataset to provide content-based movie predictions. A Pearson-r correlation is then used to recommend movies liked by other similar users.

Result: 4% improvement on absolute error when using hybrid approach as opposed to just collaborative filtering.

MovieFinder.com is a commercial system that recommends movies. Previous interests are recorded using a 5-point feedback scale, and new recommendations based on the

average customer rating. Individual movies also contain a textual prediction when they are browsed.

MovieLens [rashid02] is a movie recommender system. The movies are held in a database along with data on film genre etc. Users review movies and share this feedback with others for collaborative filtering purposes. A simple popularity/entropy recommendation strategy was used live in experiments. This project is ongoing and the MovieLens dataset is shared for other systems to use as a benchmark dataset.

Nakif [funakoshi01] is a movie recommender, using the MovieLens dataset of movie information. An incremental adaptation of the Pearson-r correlation algorithm is applied to user ratings within the dataset.

Results: 0.7 precision, 1.0 recall on the MovieLens dataset.

Recommendation Explorer [efron01] is a film recommender system, using the reel.com movie dataset. It uses latent semantic indexing (LSI) to discover interesting movies from an initial seed of 10 or so movie ratings.

Results: 0.1 precision, 0.75 recall on the reel.com dataset.

Reel.com is a commercial system that recommends movies based on customer reviews. The customers enter their movie requirements (genre, viewing format, price etc.) and a set of recommendations is computed based on the habits of other customers.

Tatemura's [tatemura00] system uses virtual reviewers for recommendation of movies. Users can explicitly rate movies they have seen. Users can collate movies together to form a new viewpoint (a virtual reviewer), and ask for recommendations from this viewpoint. A vector-space model is used to compute the similarity between a particular viewpoint and known movies, and a scatter/gather method used to navigate this space.

Music domain

CDNOW is a commercial music CD shop/recommender system. Customers provide feedback as to which artists they prefer and own. Likes and dislikes can be indicated

and a set of 6 albums recommended upon request. Feedback on these recommendations is also elicited.

A standard search facility is provided, and 10 other related albums to any single album recommended. A list of albums the customer owns is maintained, and new purchases are added to this list.

CoCoA [aguzzoli01] is a music recommender system. As users add and delete tracks for their own music compilations, the recommender system will suggest potential tracks. Case-based reasoning is used to classify tracks into genres, and a pearson-r correlation used to find other people who have similar tastes to this user. Recommendations are then based on a cosine similarity measure of those track compilations liked by the set of similar people.

Results: 0.563 precision on Medline dataset.

Ringo [maes94] is a music recommender system. It uses collaborative filtering based on user's ratings of music albums. Virtual users are created to bootstrap the system, providing some initial stereotypical ratings (e.g. a virtual Madonna fan). Pearson r correlation coefficients are used to determine similarity.

Results: A real/predicted scatter plot is presented

News filtering domain

GroupLens [konstan97] collaboratively recommends Usenet newsgroup articles. A ratings database containing user ratings for each message, and a correlations database containing pairs of similar users, is maintained. A Pearson correlation algorithm was used to find similar users (applied within single newsgroups since ratings were too sparse to work for all newsgroups).

Results: Various Usenet use figures are presented.

PHOAKS [terveen97] recommends web references found in Usenet news articles. A hand-crafted set of filter rules is used to classify web resources into categories. Web references are then given a rating based on the number of authors that recommend the reference (the idea being frequently referenced web pages are good ones). Each newsgroup thus has a set of ranked recommendations to web pages.

Results: The filter rules have a precision of 88% with a recall of 87%. When compared to the newsgroups FAQ, the 1st place URL had a 30% probability of appearing in the FAQ.

P-Tango [claypool99] is an on-line newspaper article recommender. Users provide explicit keywords of the type of articles they are interested in and the system recommends other articles that may be of interest. A separate Pearson r correlation and content-based overlap coefficient method is used, using a dynamically weighted combination function to choose between methods of recommendation.

Web domain

Community Search Assistant [glance01] is a meta-search engine, which maintains a database of previous search queries from which to recommend. No user feedback is required, as similar search queries are identified using heuristics that find similarity within the search graphs. User search queries are shared. Similarity is based on query keyword correlation.

Fab [balabanović97] recommends web pages based on relevance feedback from users. User's rate recommended web pages on a 7-point scale. A set of collection agents dynamically formulates useful groups of pages, with successful agents duplicated and unsuccessful agents removed (success is a measure of feedback scores). Several agent heuristics are used to create page groups, including using commercial search engine results, random picks and human selected "cool" sites.

Selection agents pick pages from the collection agent topics for recommendation (thus sharing topics between users). A profile is built from the terms of the pages (selection agents have user profiles, collection agents have topic profiles).

Results: ndpm measure (distance of user rankings from profile rankings) 0.2 - 0.4 using Fab system, 0.75 – 0.5 using random selection

ifWeb [asnicar97] assists web navigation and recommends pages similar to example pages provided by the user. A vector-based user profile is maintained using features of web pages (host, size etc.) and a semantic network for term page co-occurrence relationships. The users provide explicit feedback on page relevance (positive and negative). A temporal decay function weights features within the user model, which is

represented using UMT. A tree interface can be displayed showing where the current web page is located in relation to its links, and how interesting the links are from it (based on correlation between the crawled pages and the user profile). A search for similar documents is also available, using the same mechanism.

Results: Results on tests using 4 subjects on a limited set of documents (4-6). 9 sessions were conducted, with learning from feedback occurring between each session. Precision 65%, ndpm 0.2

MEMOIR [de roure01] records user web navigation trials and provides a framework for recommendation of web pages. Users must manually enter URL trials into the system, which are then shared with all users. MEMOIR monitors user web browsing and tries to correlate the current web position with a trial. Web pages (based on trial end points) are recommended when a correlation is made. In addition to web pages, similar users can be recommended based on keyword profiles derived from their trials.

METIOREW [bueno01] is a web page recommender. A user model is created for each user detailing pages visited and feedback provided. Profile keywords are extracted from relevant web pages and stored in A TF representation. New pages are classified for relevance using a naïve Bayes classifier. A Pearson-r correlation is used to find similar users and thus discover people with similar objectives, allowing keywords to be shared.

ProfBuilder [wasfi99] monitors web site use and recommends pages from that site to new visitors. A vector-space user profile is constructed from the pages a visitor has seen so far. The content of the pages make up the vectors using TF-IDF weightings. Stemming and stop words are used to reduce vector dimensionality, and a vector cosine measure used to measure vector similarity. For each page in the web site, the probability of previous users moving down a link is computed from historical navigation patterns. Both similar pages and pages historically likely to be navigated from the current page are selected for recommendation.

QuIC [el-beltagy01] recommends web pages based on the currently browsed page. A set of link-bases are created and shared by users. A custom clustering algorithm clusters sets of web pages represented in a bag-of-words format using TF-IDF term

weighting. These clusters are associated with the pages in the link-bases. The users currently browsed web page is then matched to a cluster and the most relevant link-base of web pages recommended.

RAAP [delgado98] recommends web pages. It uses a multi-class profile representation, but classes are not shared among users. Users explicitly rate pages as interesting, and provide a class in which to place these pages. Negative examples are inferred by monitoring deletion operations or provided explicitly. A TF-IDF document weighting is used along with a modified Rocchio classifier. Stemming, stop list and an information gain measures is used to reduce term dimensionality. Pearson-r correlation is used to find similar users upon which to base recommendations.

Siteseer [rucker97] recommends web pages collaboratively. Bookmarks are used to find similar users (by computing the overlap of a user's bookmarks with the other users' bookmarks). Recommended URLs thus derive from the bookmark lists of similar users.

Results: 1000 users, 18% confidence recommending 88% of the time.

SOAP [voss97] is a multi-agent system that recommends web sites. User, search and recommender agents communicate to achieve recommendation for multiple users. Users can submit queries to an agent, which calls a search engine. The search results are associated with the query (taken as the "topic" of the resulting URLs). Users can explicitly rate pages using a 5-point scale and can provide free-text annotations. User bookmarks are used to infer interest in URLs too. Recommender agents use the topic (query) and rating to filter known URLs and hence provide recommendations. Since annotations and ratings are shared, any user can inspect them. Page content is represented using keyword vectors.

SurfLen [fu00] monitors user browsing and recommends web pages. User browser history logs are mined for association rules using the A-Priori algorithm. These rules associate URLs with other URLs. When a user opens a known URL, the associated URLs are immediately recommended.

Results: Some quantitative figures for 100 simulated users (based on Yahoo log data)

Other domains

Campiello [grasso99] is a recommender system for leisure activities in a local community. Campiello elicits feedback on leisure events and places using postcard type forms. Freeform textual comments and scaled ratings are recorded. Recommendations can be requested on particular events and places, and both contend-based and collaborative recommendation is used. The Pearson algorithm is used to find similar users.

An internal database is maintained for a particular city, with a newscard reader installed at the leisure facilities (such as within museums) to accept feedback and provide recommendations.

ELFI [schwab00] recommends research funding program information to users. Users are monitored as they use the system, and positive examples obtained from observations of the type of thing they are interested in. This training set is applied to both a simple Bayes classifier and k-nearest neighbour (kNN) classifier. Funding information is held as feature vectors, and univariate significance analysis used to reduce vector dimensionality. The classifiers are used to measure the similarity of unseen database entries to the interesting training set. The closest matching pages are recommended to the user.

Results: 220 users, divided into 5 groups. The user activity logs were used as training/test data using a cross validation method. simple Bayes classifier 91-97% accuracy, kNN 94-97% accuracy

OWL [linton99] recommends new Word commands that other users have used before. User's are monitored as they use Word and behaviour log files extracted. A statistical method is used to recommend un-used Word commands that others have used.

Appendix B The Quickstep system design

This section details some of the design documents generated as part of creating the Quickstep system.

Nine independent processes make up the Quickstep system, all interacting to provide the recommendation service to its users. Figure B.1 shows the process map, detailing

how these 9 processes (the ellipses) interact. Basically, each process reads information from a data store, processes it and stores the results to another data store. File locking prevents multiple processes corrupting a data store. Processes are timed to run at periods during the day and night, and sequenced so that the results of one process (e.g. classification process) will be ready for the next process (e.g. profile builder).

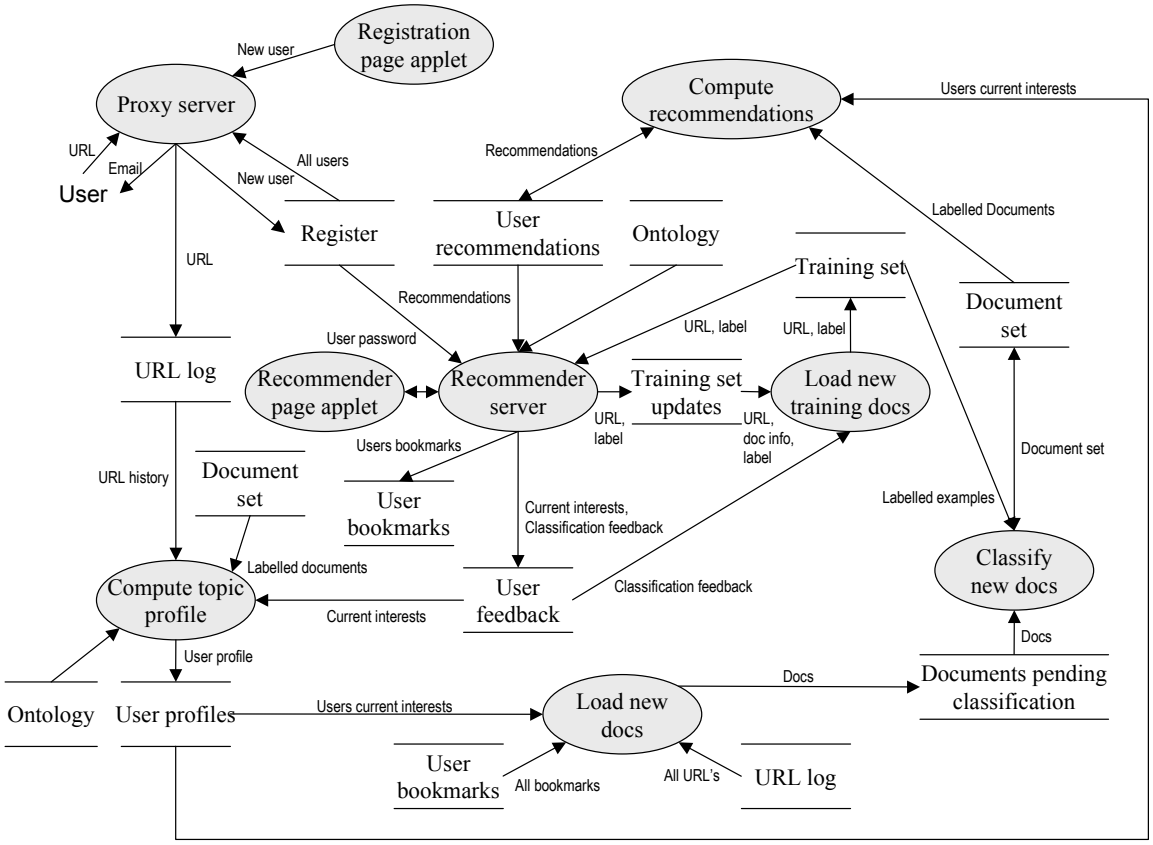


Figure B.1 : Quickstep process map

Each process is functionally decomposed in the detailed design.

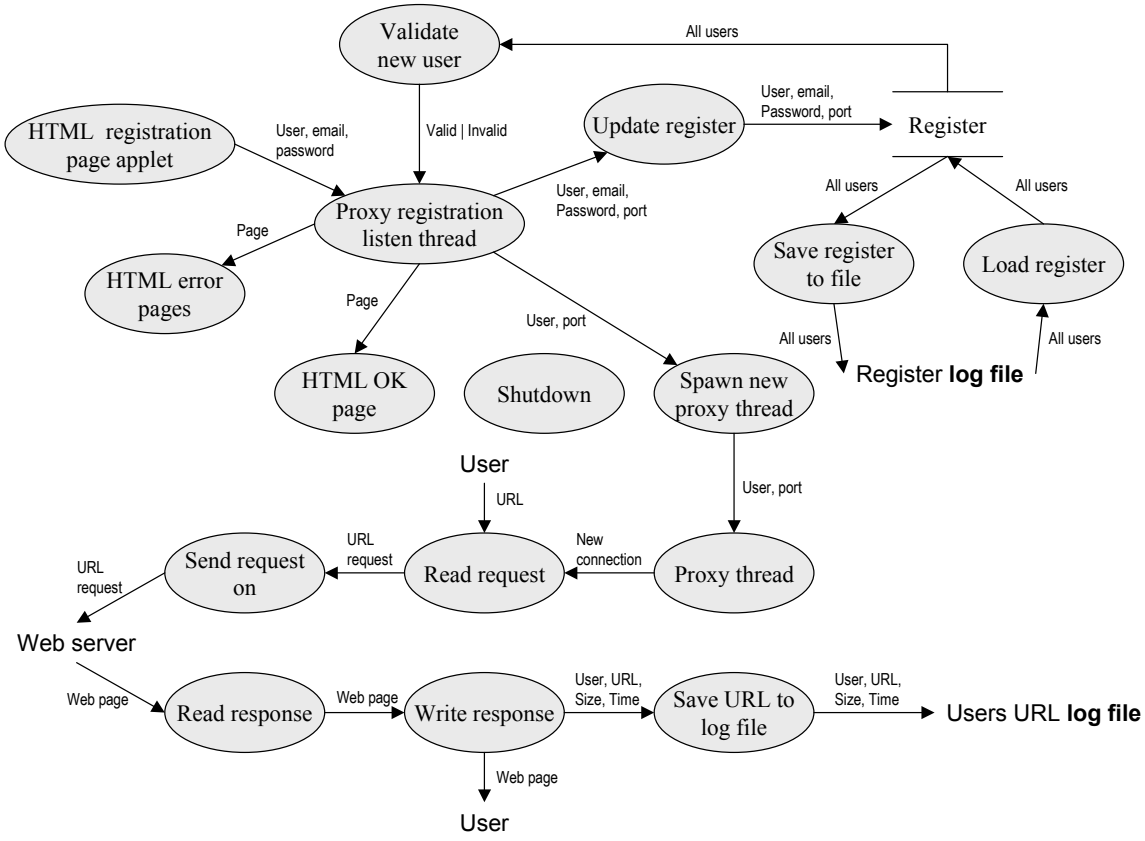


Figure B.2 : Proxy server design

The proxy server process handles requests by users for web pages. The web pages are retrieved from the web without any modification, and the URL request logged (with a time stamp). An on-line registration service is also supported, so users can register with the Quickstep system and allocate a port number. The system emails users their individual proxy port number, allowing them to enter the correct proxy server configuration details into their browsers. Once allocated, a thread is spawned to handle the new port connections. Figure B.2 shows the proxy server design.

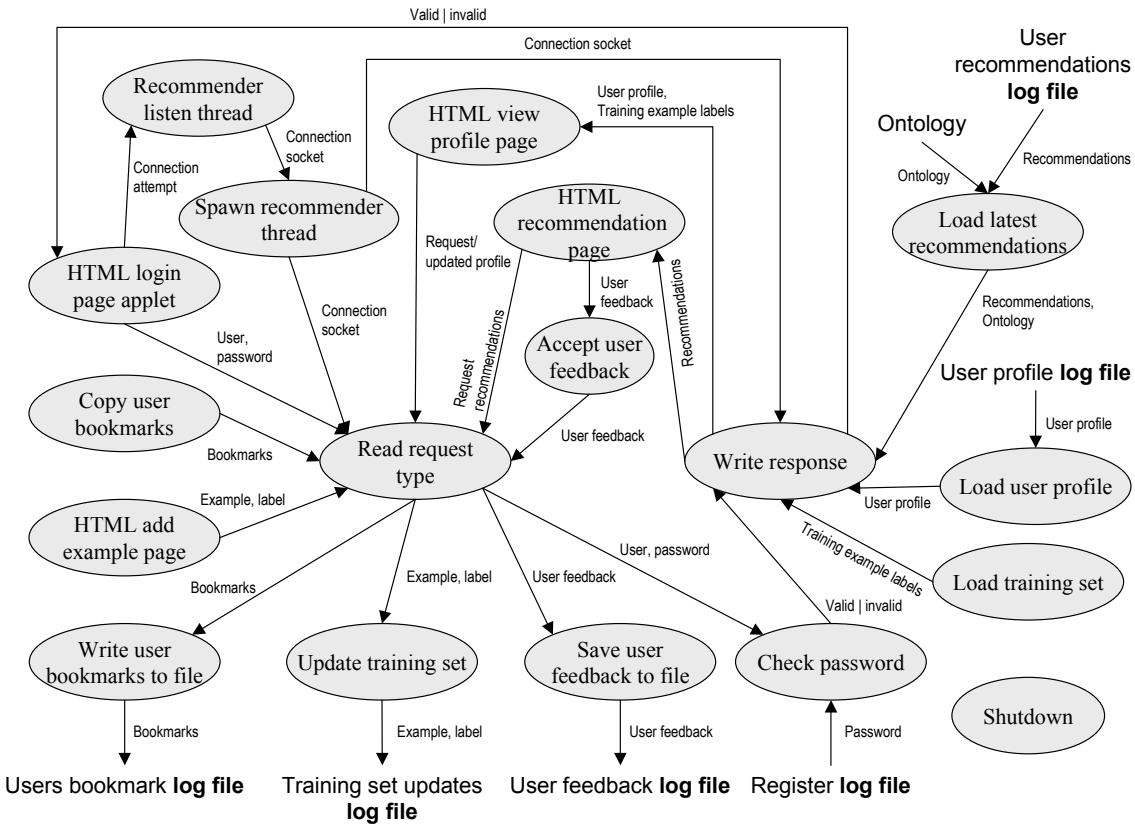


Figure B.3 : Recommendation server/applet design

The recommender server process (and its associated recommender applet process) allows the user to access their pre-computed recommendations. The users will load the applet via a web page, and it will attempt to connect to the recommendation server. A client-server set-up is required to overcome Java security restrictions, since file access is needed on the host machine to read the recommendation log file. Once the user has logged on (and the recommendation server authorised the logon), the recommendations for that user will be sent to the recommendation applet for display. These recommendations can be examined via the interface and feedback provided. When the user logs out (or closes the applet by closing the browser) the feedback is sent to the recommendation server and saved in the feedback log file. Figure B.3 shows the recommendation server/applet design.

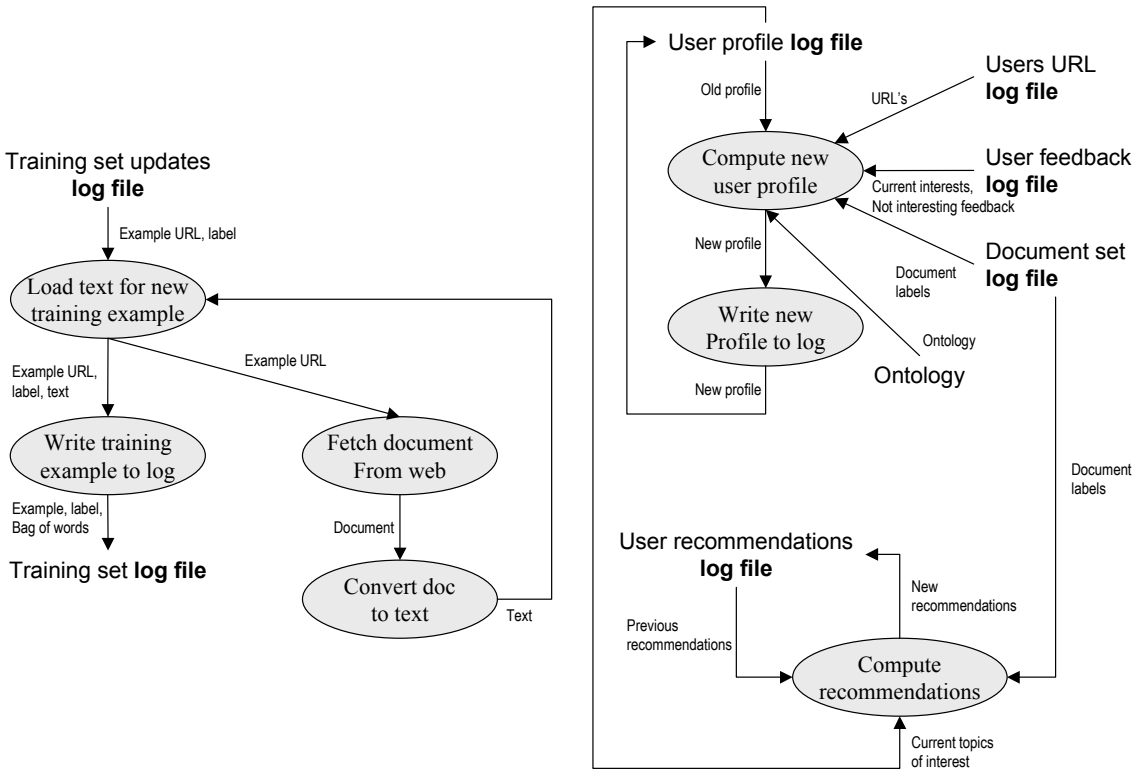


Figure B.4 : Training set compiler/profiler/recommender design

The training set compiler process loads the training set update log (created as part of the feedback saved by the recommendation server) and loads each document. The loaded documents are then saved into the training set log. The profile compiler loads the classified document store and feedback logs, and computes a profile for each user. The URL logs are correlated with the classified documents to build a time log of topics browsed. Interest values can then be calculated from this topic history and the feedback logs. The recommendation compiler loads the user's profiles and correlates them with the classified documents to build a ranked list of potential recommendations. The top 10 recommendations are stored in the user's recommendation log. Figure B.4 details the design of these processes.

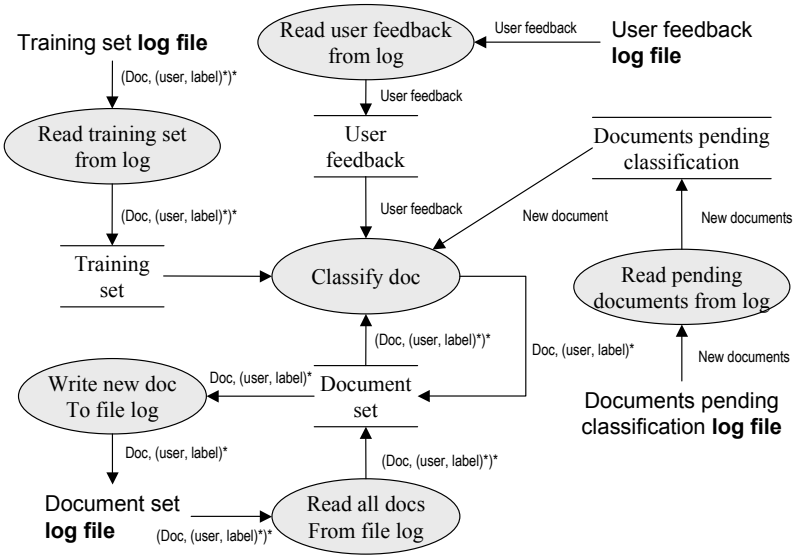


Figure B.5 : Classifier design

The classifier process loads the current training set and builds a new classifier each time it is run. It then iterates over the pending document store and classifies each one. These classified documents are then moved to the classified document store. This is a slow process, so is run overnight. Figure B.5 details the classifier process.

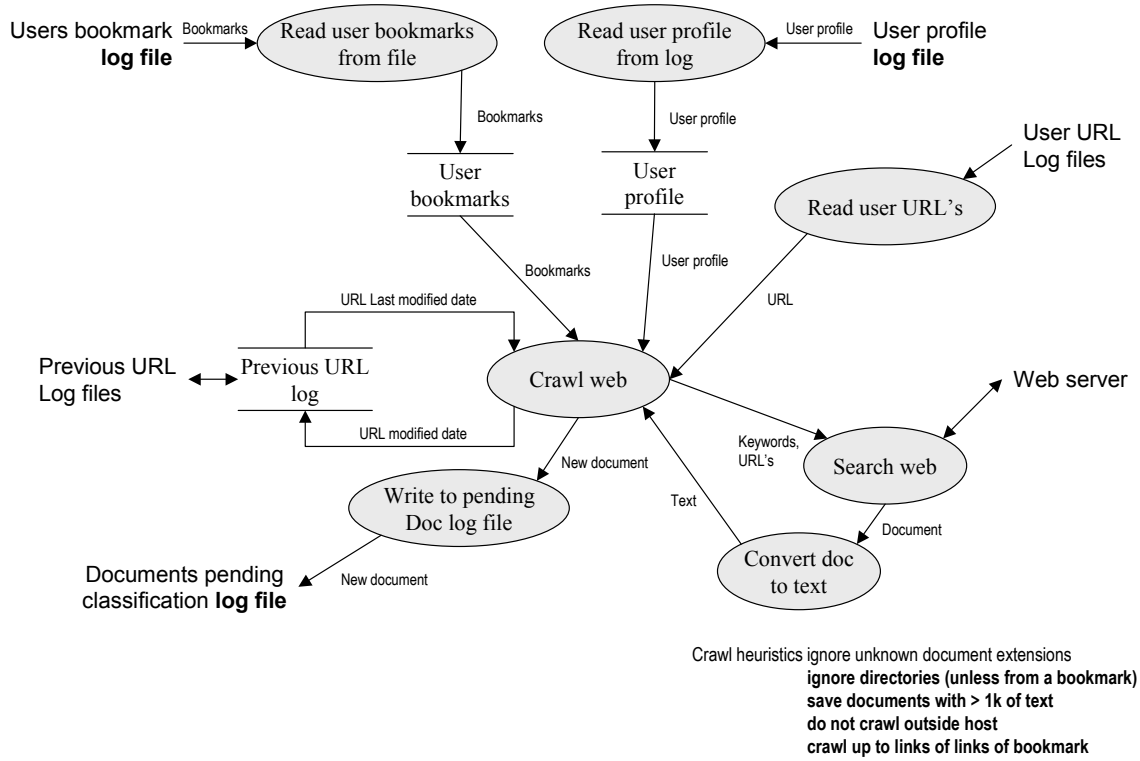


Figure B.6 : Web crawler design

The last process is the web crawler. This loads new documents from the web by crawling bookmarks (loaded into the system at the start) and loading each URL found

in the user's URL log files. Only PS or PDF documents are kept (compressed versions are decompressed and used too). The documents kept are saved in the pending document store, ready for the classifier to handle them. Figure B.6 describes the web crawler process.

URL log	(date <tab> user <tab> size <tab> URL <newline>)*
(1 file per user)	/URLlogs/<user>_url.log
Register	(user <tab> password <tab> email <tab> port <tab> start date <newline>)*
	/Register.log
	/RegisterBackup.log
User recommendations	(user <tab> date <tab> <type> <tab> title <tab> URL <tab> <topic list> <tab> confidence <newline>)*
(1 file per user)	<type> = "recommendation" | "topic list"
	<topic list> = "(" (topic <tab>)* topic ")"
	/Recommendations/<user>_recommend.log
User feedback	(user <tab> date <tab> "classification info" <tab> URL <tab> (topic",")* topic <newline>)* |
(1 file per user)	(user <tab> date <tab> "interest rating" <tab> URL <tab> interest rating <newline>)* |
	(user <tab> date <tab> "URL jump" <tab> URL <newline>)* |
	(user <tab> date <tab> "current interests" <tab> (topic <tab>)* topic <newline>)*
	/Feedback/<user>_feedback.log
User profiles	(user, (current interests)*, (interest history)*, system data)*
	/Profiles.log
User bookmarks	(bookmark <newline>)*
(1 file per user)	/Bookmarks/<user>_bookmarks.log
Training set	URL, date, (user, topic)*, bag of words
(1 file per URL)	/TrainingSet/<doc id>.arff
Training set update list	(URL, date, (user, topic)*)*
	/TrainingSet/UpdateList.log
Document set	URL, date, last modification date, (topic)*, bag of words
(1 file per URL)	/DocumentSet/<doc id>.arff
Documents pending classification	(URL, date, bag of words)*
(1 file per URL)	/PendingDocuments/<doc id>.arff
Previous URL's	(URL <tab> date_last_checked <newline>)*
	/PreviousURLs.log
Users URL log checklist	(user <tab> date_checked_up_to <newline>)*
	/WebSearchChecklist.log

Figure B.7 : Quickstep log files

The log file formats are also detailed here. This gives some idea of what sort of data is being stored in each of the log files. Figure B.7 shows this.

Appendix C The Foxtrot system design

This section details some of the design documents generated as part of creating the Foxtrot system.

Nine independent processes make up the Foxtrot system, all interacting to provide the recommendation service to its users. Figure C.1 shows the process map, detailing how these 9 processes (the circles) interact. Basically, each process reads information from a data store, processes it and stores the results to another data store. File locking prevents multiple processes corrupting a data store. Processes are timed to run at periods during the day and night, and sequenced so that the results of one process (e.g. classification process) will be ready for the next process (e.g. profiler).

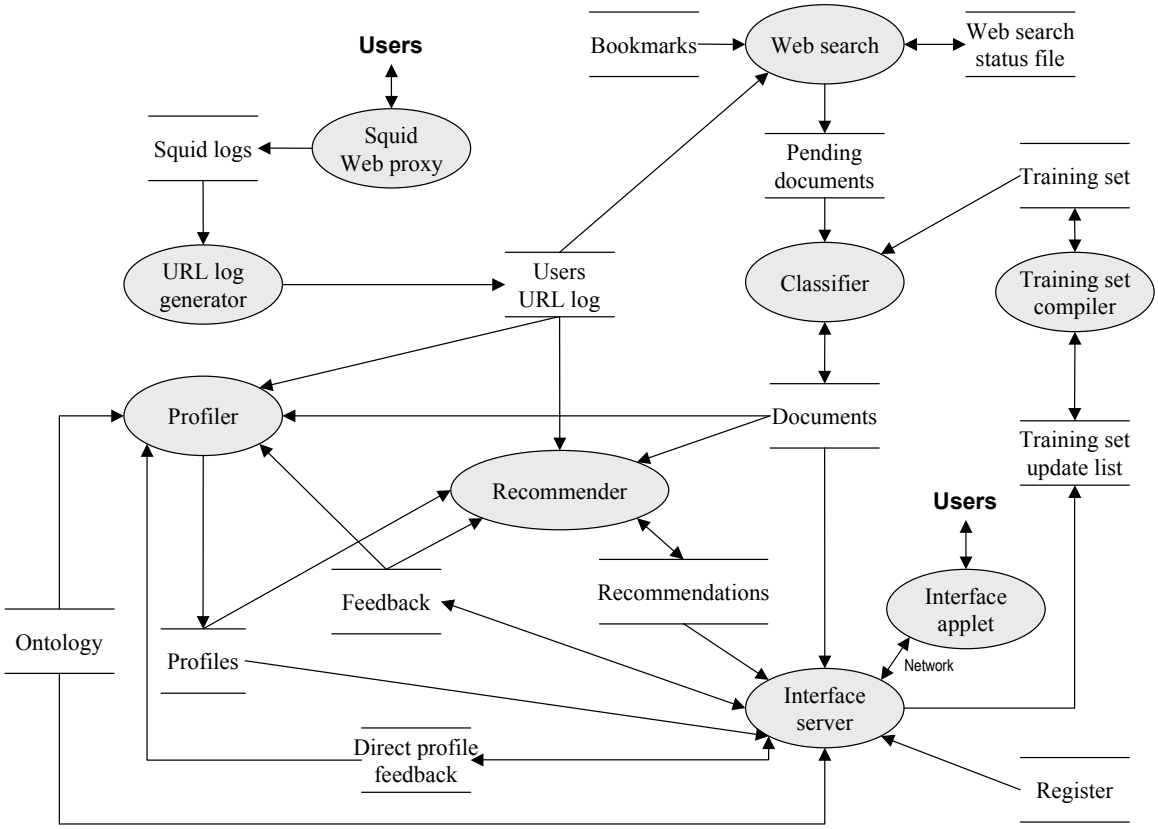


Figure C.1 : Foxtrot process map

Each process is functionally decomposed in the detailed design.

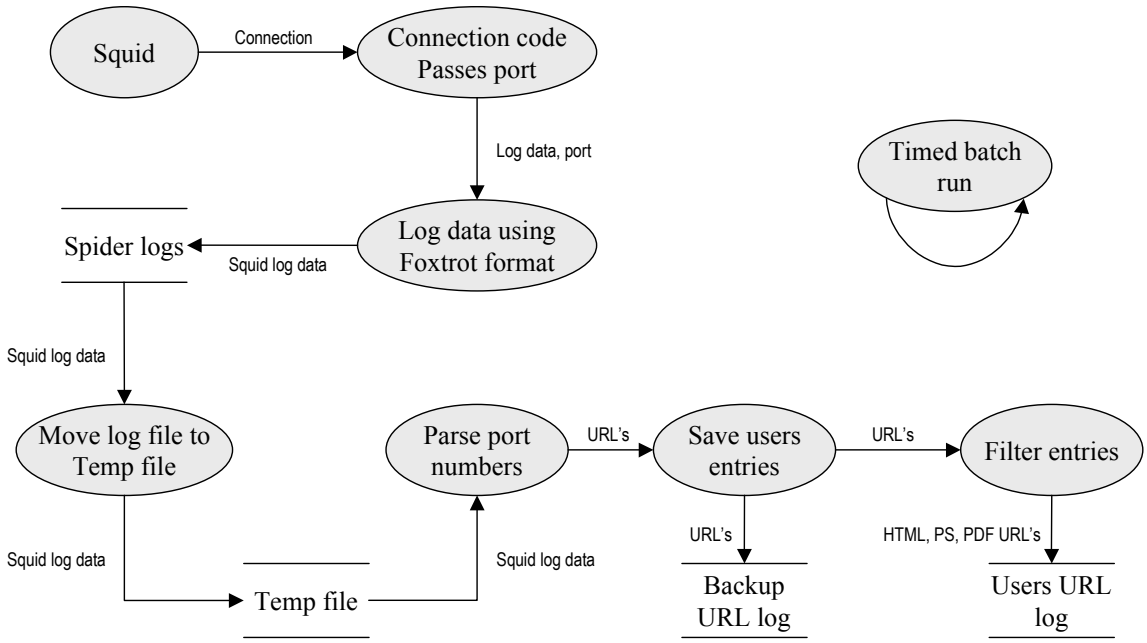


Figure C.2 : Squid web proxy and the URL log generator design

The Squid web proxy process handles requests by users for web pages. Squid is a well-respected third party web proxy, written in C. It is open source and supports the latest HTTP protocols. Foxtrot will use a modified version, logging port numbers with the user's normal URL logging information. As Squid rotates its log files a second URL log generator process will regularly copy the log files, parse them and save each URL log entry to a separate user log file. Figure C.2 shows these two processes.

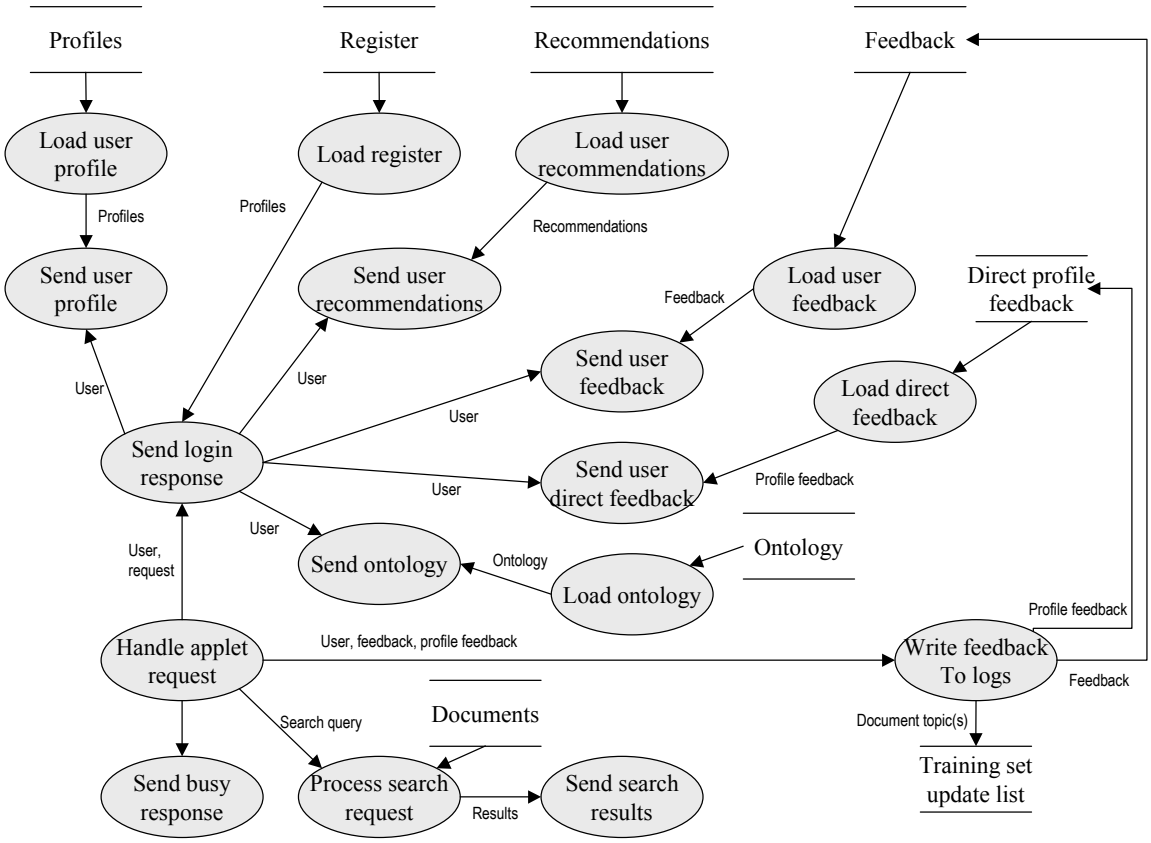


Figure C.3 : Interface server design

The interface server handles requests from the interface applet for recommendations. The client/server set-up is needed to overcome security restrictions applied by most browsers on applets. Once a login request is received the user's recommendations, profile and the current topic ontology are sent to the applet. Upon logout, any user feedback is appended to the feedback log (and any topic labels added to the training set update list). The server handles search requests by reading a search query and spawning a thread to handle it. The applet will regularly check on the status of the search query and will be sent a busy signal until it is complete. When ready, the search result (a set of URLs) is returned. Figure C.3 shows the interface server process.

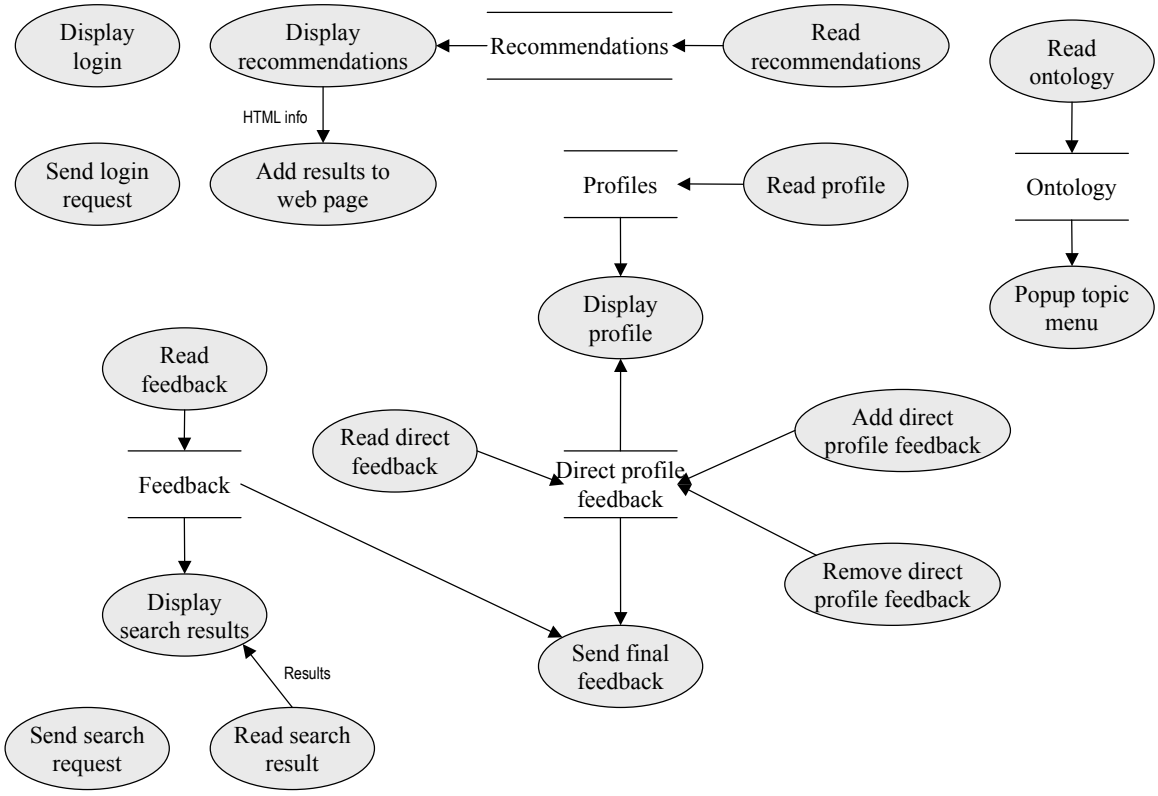


Figure C.4 : Interface applet design

The interface applet is run in a web page, and is Foxtrot's interface to the users. Users must first logon, where they are presented with a search interface and a set of recommendations. These recommendations are sent by the interface server, and are the URLs Foxtrot thinks will be most interesting to the user. These recommendations can be examined, and feedback provided (paper quality, paper topic interest, corrections to paper topic). A popup menu allows topic corrections to be effected. The user can visually see their profile (if they are in the subject group with this feature enabled) and provide absolute reference points for the interest graph; this is the direct profile feedback.

The major functionality of Foxtrot to the users is the search system. Users can enter keywords for title search, topics for category search etc. The final search query is sent to the interface server and processes. The applet regularly checks with the interface server to see if the search query has been handled. A busy indication (such as the hourglass and a "searching…" message) is displayed until it has, when the full search results are shown (a list of ranked URLs).

Upon logout the user's feedback is sent to the interface server. Figure C.4 shows the interface applet design.

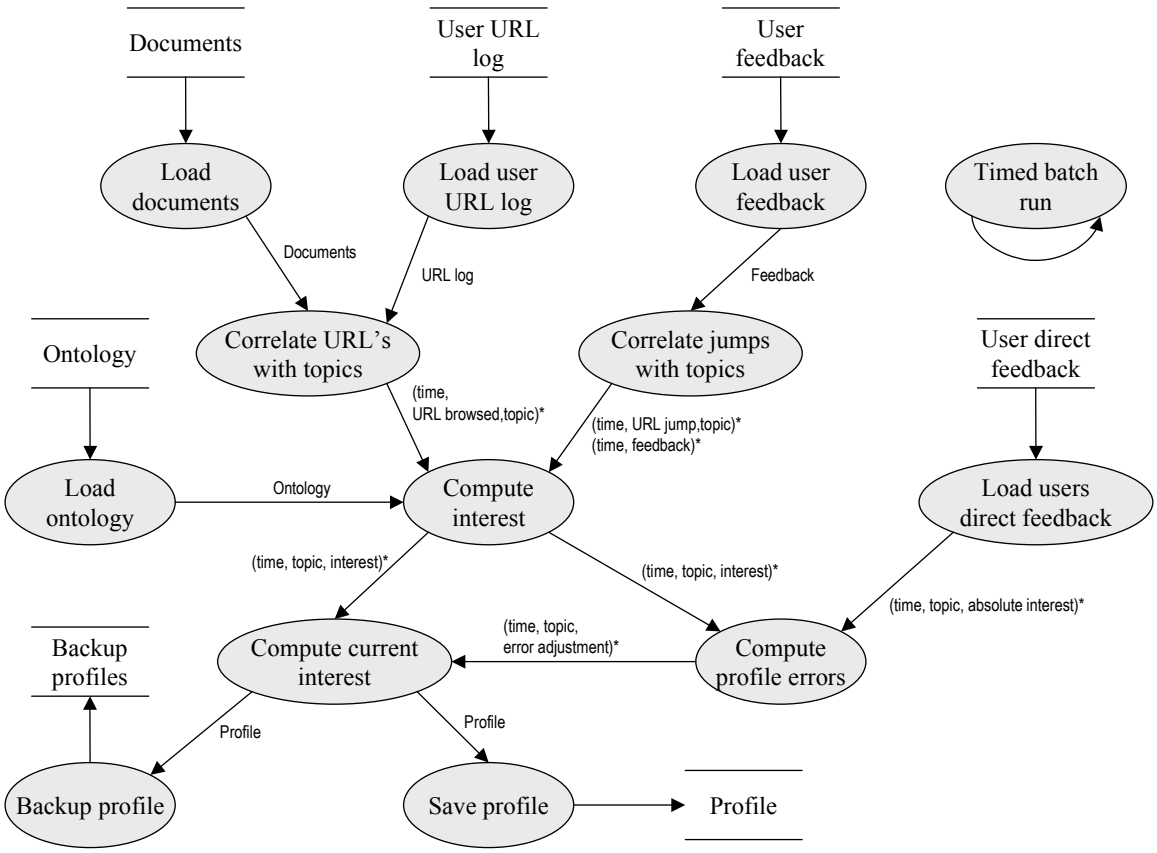


Figure C.5 : Profiler design

The profiler reads the URL logs generated by the web proxy, and correlates them with the classified documents to formulate an interest event time-line. Feedback events are also added as interest events. The interest events are then used to formulate a profile via a time-decay function. If any direct profile feedback exists, this is also used to improve the profile. Newly computed profiles overwrite the existing ones with a backup made for later analysis work. Figure C.5 shows the profiler design.

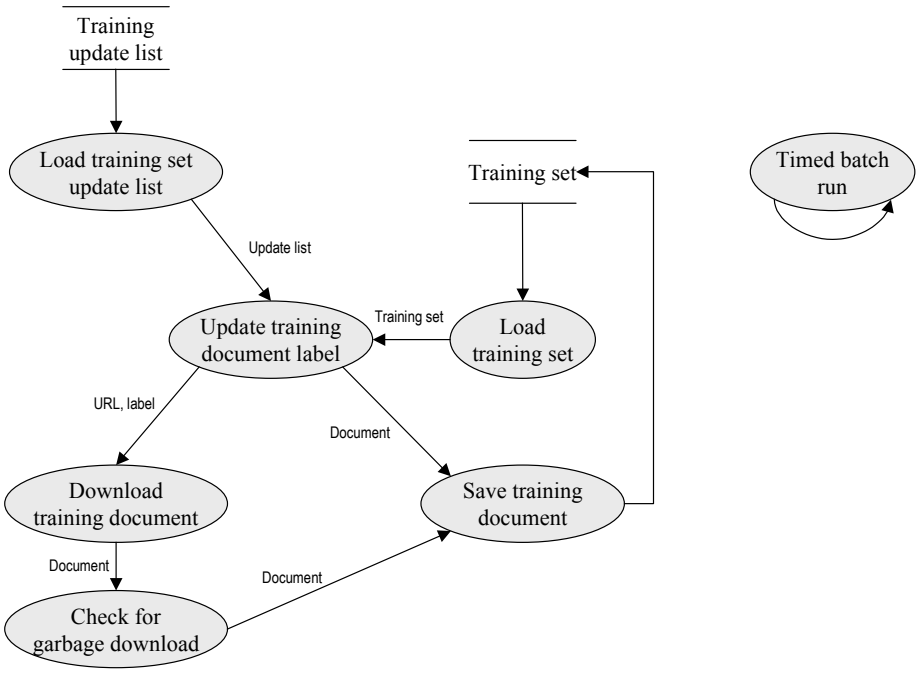


Figure C.6 : Training set compiler design

The training set compiler reads from the training set update list and downloads any URLs found. They are then given a label and saved in the training set. If the document exists already in the training set its label is updated. Figure C.6 shows the training set compiler design.

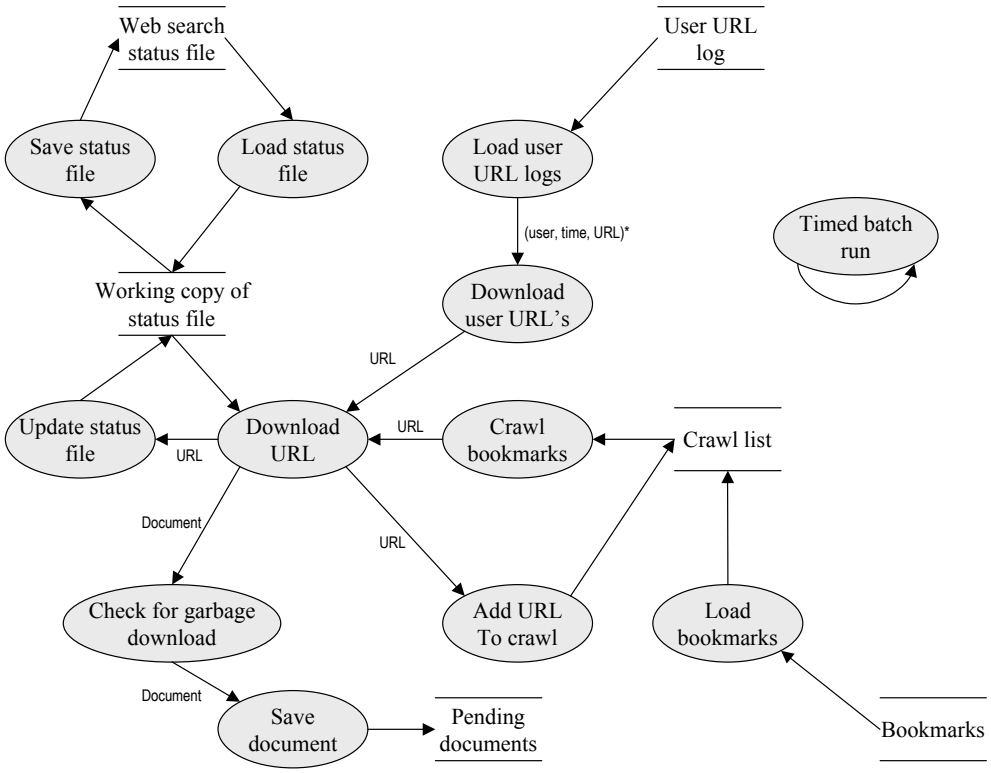


Figure C.7 : Web search design

The web search process loads all URLs browsed by the user and adds them to the pending set. They will be loaded later and classified by the classifier process. A set of bookmarks are also crawled and downloaded if changed. A status file (which includes document links and the last modified date) is kept to avoid re-loading documents that have not changed. A garbage check is made before keeping downloaded documents as the text conversion utility can sometimes fail and produce garbled words. Figure C.7 shows the web search design.

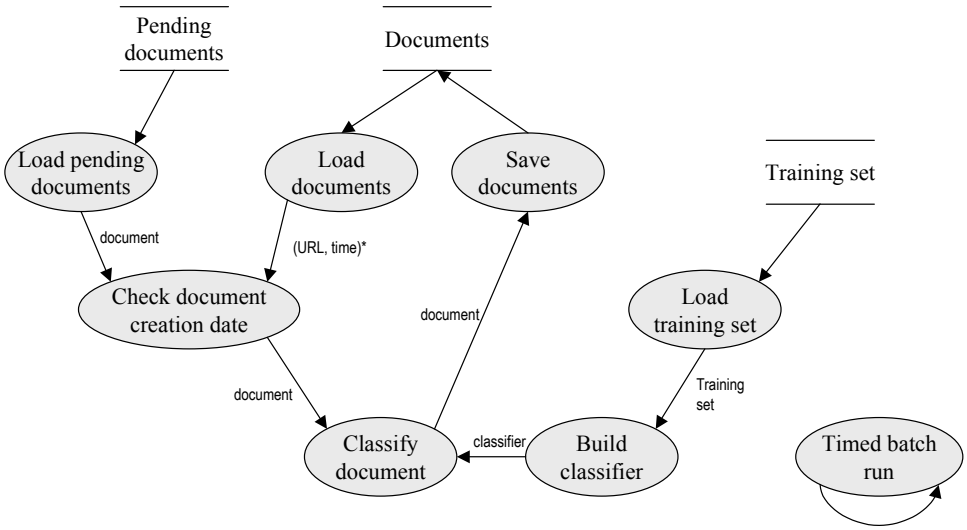


Figure C.8 : Classifier design

The classifier process loads all documents in the pending set and attempts to classify them. The current training set is used to build the classifier. Once built the classifier is applied to all pending documents and labels generated. The labelled documents are added to the document set. Figure C.8 shows the classifier design.

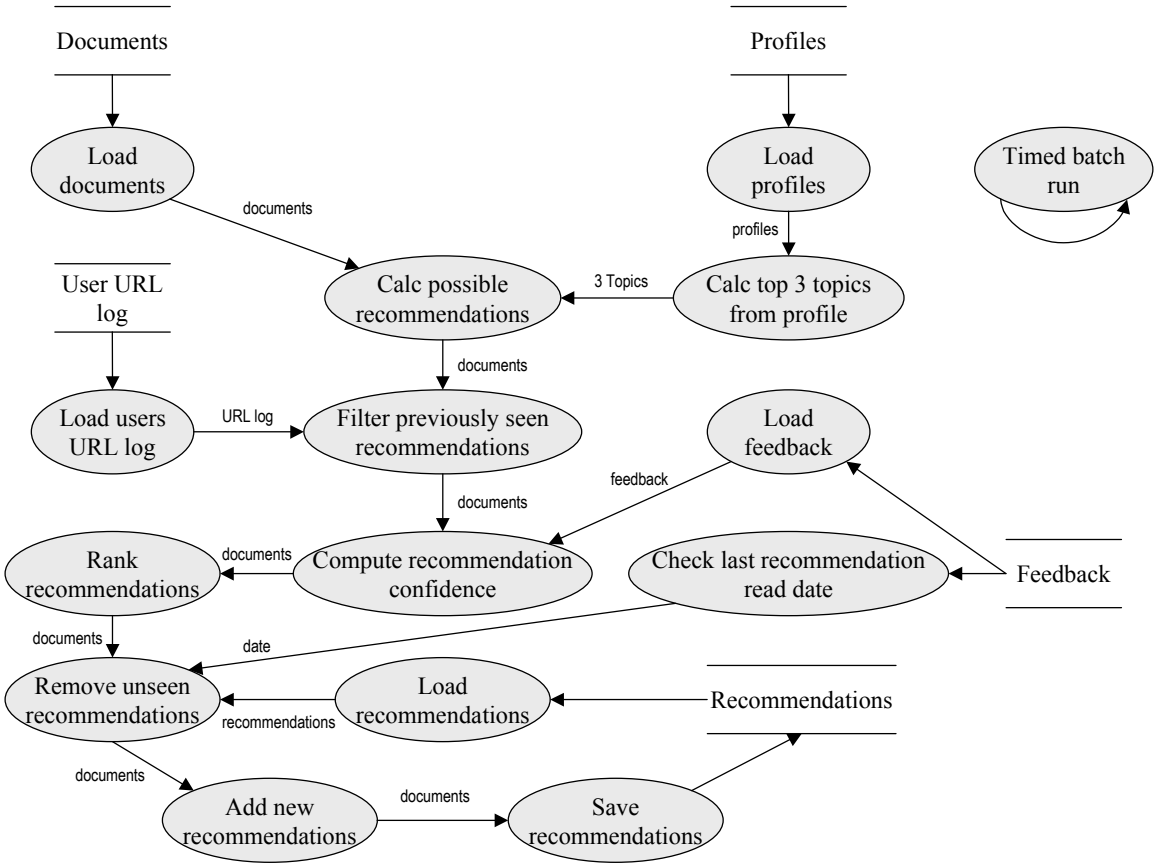


Figure C.9 : Recommender design

Lastly we have the recommender process, which takes the user profiles and formulates a set of recommendations for each user. The top three topics are extracted from the user's profile and correlated with the classified documents. Previously seen documents are removed so that the same thing is not recommended twice. The remaining recommendation set is ranked by recommendation confidence (based on quality feedback and classification confidence). The existing recommendation set is then pruned for any unread recommendations and the new set appended. Thus, the next time the user opens the interface applet this recommendation set will be read. Figure C.9 shows the recommender design.

Glossary of machine-learning terms

For a more detailed description of machine-learning, [sebastiani02] provides an excellent overview of machine-learning techniques, as does [mitchell97].

A-Priori algorithm – An optimisation algorithm for reducing the number of large itemsets. Used in data mining (for example when finding association rules).

Backpropagation - Neural network algorithm for updating hidden layer weights. A reliable technique, it is the backbone of many neural networks.

Bag of words – Document representation consisting of a list of words and the number of times the words appear (term frequency).

C4.5 – ID3 variant, applying rule post-pruning and other additional techniques.

Cosine similarity – dot product measure of the distance between two vectors. This is used to measure similarity between two documents when the vector space represents document features.

Decision tree – Algorithm using a tree, with each node of the tree dividing the hypothesis space using an attribute. As the tree is traversed, from top to bottom, the hypothesis space is increasingly sub-divided until only one hypothesis is left. Decision trees can be easily converted into classification rules.

Entropy – A measure of the "purity" of a collection of examples. It measures the difference between the number of positive and negative examples (zero is a "pure" or perfectly balanced set).

FAQ – Frequently Asked Questions – A document with often asked questions answered aimed at helping novice users.

ID3 – Classic decision tree learning algorithm. Uses information gain to select node terms.

ID4 – Variant on ID3.

Information gain – Measure of the expected reduction in entropy of a term.

Jaccard coefficient – No. of pages containing two entities / no. of pages with either entities

Keyword vector – A vector of keywords. Vector has length equal to the number of terms in a document set, and values are the frequency of each term (usually applied to a document to give a document vector).

LSI – Latent semantic indexing [deerwester90]. A term weighting algorithm used for dimensionality reduction, meant to bring out the "latent" semantic structure of the vocabulary used in a document corpus.

Naïve Bayes classifier – Probabilistic classifier based around Bayes theorem. Term probabilities are assigned to classes, and for a new document the probability of belonging to any particular class is computed.

Nearest neighbour – Learning algorithm that measures the distance between document vectors within a vector-space representation. The distance indicates similarity of documents (the nearest neighbours) – cosine similarity is often used.

Neural network – Network of units, with inputs usually representing terms and outputs classes. Connections between units have weights, which are trained by loading examples (using a training rule such as backpropagation to update weights).

Pearson r correlation – Type of information measure, used to weight terms with respect to positive and negative examples.

Reinforcement learning – Learning algorithm where actions produce rewards or penalties, thus the most rewarding sequence of actions is reinforced (hence learnt).

Rocchio classifier – Learning algorithm, often used with TF-IDF weightings. Class term vectors are computed by summing positive example weights and subtracting negative example weights.

SMART – An indexing engine, which converts documents into document vectors. It uses TF-IDF weighting.

Stemming – Removal of suffixes from words. Used to reduce the number of terms that are synonyms in a textual document.

TF – Term frequency. The number of times a term (often a word or phrase) occurs within a document.

TF-IDF – Term frequency – Inverse Document Frequency. Algorithm for assigning weights to terms in a document set, biased to weight the most discriminating terms highest.

References

[agrawal94] Agrawal, R. Srikant, R. "Fast Algorithms for Mining Association Rules", Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, 1994

[aguzzoli01] Aguzzoli, S. Avesani, P. Massa, P. "Compositional Recommender Systems Using Case-Based Reasoning Approach", 2001 ACM SIGIR Workshop on Recommender Systems, 2001, Radisson Hotel New Orleans, LA - USA

[aha91] Aha, D. Kibler, D. Albert, M. "Instance-based learning algorithms", Machine Learning, 6:37-66, 1991

[akt-ontology] http://www.aktors.org/publications/ontology/

[alani02] Alani, H., O'Hara, K., and Shadbolt, N. "ONTOCOPI: Methods and Tools for Identifying Communities of Practice", Intelligent Information Processing Conference, IFIP World Computer Congress (WCC), Montreal, Canada, 2002

[albert02] Albert, R. and Barabasi, AL. Statistical Mechanics of Complex Networks. Review of Modern Physics, 74, 47, 2002

[arslan02] Arslan, B. Ricci, F. "Case-Based Session Modeling and Personalization in a Travel Advisory System", Workshop on Recommendation and Personalization in e-Commerce (RPeC02), Malaga, Spain

[asnicar97] Asnicar, F. A. Tasso, C. "ifWeb: a Prototype of User Model-Based Intelligent Agent for Document Filtering and Navigation in the World Wide Web", In Proceedings of the Sixth International Conference on User Modeling, Chia Laguna, Sardinia, June 1997

[balabanović97] Balabanović, M. Shoham, Y. "Fab: Content-Based, Collaborative Recommendation", Communications of the ACM 40(3), March 1997, 67-72

[bauer02] Bauer, M. Dengler, D. "Group Decision Making Through Mediated Discussions", Workshop on Recommendation and Personalization in e-Commerce (RPeC02), Malaga, Spain

[becerra-fernandez00] Becerra-Fernandez, I. Facilitating the Online Seach of Experts at NASA using Expert Seeker People-Finder. Proceedings of the 3rd

International Conference on Practical Aspects of Knowledge Management (PAKM), Basel, Switzerland, 2000

[billsus98] Billsus, D. Pazzani, M. J. “A Personal News Agent that Talks, Learns and Explains”, In Autonomous Agents 98, Minneapolis MN USA

[bollacker98] Bollacker, K. D. Lawrence, S. Giles, C. L. “CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications”, In Autonomous Agents 98, Minneapolis MN USA

[breiman94] Breiman, L. “Bagging predictors”, Technical Report 421, Department of Statistics, University of California at Berkeley, 1994

[brown00] Brown and Duguid 2000. “The social life of information”, Harvard Buisness School Press

[shum00] Shum, S.B. Motta, E. Domingue, J. “ScholOnto: An Ontology-Based Digital Library Server for Research Documents and Discourse”, International Journal on Digital Libraries, 2000, Springer-Verlag

[bueno01] Bueno, D. Conejo, R. David, A. A. ”METIOREW: An Objective Oriented Content Based and Collaborative Recommending System”, Twelfth ACM Conference on Hypertext and Hypermedia, Hypertext 2001, Århus, Denmark

[buono01] Buono, P. Costabile, M. F. Guida, S. Piccinno, A. Tesoro, G. “Integrating User Data and Collaborative Filtering in a Web Recommendation System”, Proc. Third Workshop on Adaptive Hypertext and Hypermedia, UM2001, Sonthofen, Germany

[burke00] Burke, R. “Knowledge-based Recommender Systems”, In: A. Kent (ed.): Encyclopedia of Library and Information Systems, 2000, Vol. 69, Supplement 32.

[claypool99] Claypool, M. Gokhale, A. Miranda, T. Murnikov, P. Netes, D. Sartin, M. “Combining Content-Based and Collaborative Filters in an Online Newspaper”, Workshop on Recommender Systems: Algorithms and Evaluation, ACM SIGIR '99

[cöster02] Cöster, R. Gustavsson, A. Olsson, T. Rudström , A. “Enhancing

Web-Based Configuration with Recommendations and Cluster-Based Help", Workshop on Recommendation and Personalization in e-Commerce (RPeC02), Malaga, Spain

[de roure01] De Roure, D. Hall, W. Reich, S. Hill, G. Pikrakis, A. Stairmand, M. "MEMOIR – an open framework for enhanced navigation of distributed information", Information Processing and Management, 37, 53-74, 2001

[deerwester90] Deerwester, S. Dumais, S. T. Furnas, G. W. Landauer, T. K. Harshman, R. "Indexing by latent-semantic indexing", Journal of the American Society of Information Science 41(6), 1990, Pages 391-407

[delgado02] Delgado, J. Davidson, R. "Knowledge Bases and User Profiling in Travel and Hospitality Recommender Systems", 9th International Conference for Information and Communication Technologies in Travel & Tourism, ENTER 2002

[delgado98] Delgado, J. Ishii, N. Ura, T. "Intelligent collaborative information retrieval", In progress in Artificial Intelligence-IBERAMIA'98, Lecture Notes in Artificial Intelligence Series No. 1484, 1998

[dmoz] dmoz open directory project, Project home page http://dmoz.org/

[domingos97] Domingos, P. Pazzani, M. "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss", Machine Learning (29), 1997, 103-130

[dunlop00] Dunlop, M. D. Development and evaluation of clustering techniques for finding people. Proceedings of the 3rd International Conference on Practical Aspects of Knowledge Management (PAKM), Basel, Switzerland, 2000

[efron01] Efron, M. Geisler, G. "Is it all About Connections? Factors Affecting the Performance of a Link-Based Recommender System", Proceedings of the SIGIR 2001 Workshop on Recommender Systems, 2001, New Orleans, LA

[el-beltagy01] El-Beltagy, S. Hall, W. De Roure, D. Carr, L. "Linking in Context", Proc The Twelfth ACM Conference on Hypertext and Hypermedia, Hypertext '01, ACM ACM Press

[eriksson99] Eriksson, H., Fergeson, R., Shahr, Y., and Musen, M. (1999). Automatic generation of ontology editors. 12th Workshop on Knowledge Acquisition, Modelling, and Management (KAW'99), Ban, Alberta, Canada

[freund96] Freund, Y. Schapire, R. E. "Experiments with a New Boosting Algorithm", Proceedings of the Thirteenth International Conference on Machine Learning, 1996

[fu00] Fu, X. Budzik, J. Hammond, K. J. "Mining Navigation History for Recommendation", In Proceedings of the 2000 Int. Conf. on Intelligent User Interfaces (IUI'00). New Orleans, Louisiana

[funakoshi01] Funakoshi, K. Ohguro, T. "Evaluation of Integrated Content-based Collaborative Filtering", ACM SIGIR Workshop on Recommender Systems, 2001

[ghani02] Ghani, R. Fano, A. "Building Recommender Systems Using a Knowledge Base of Product Semantics", 2002, Workshop on Recommendation and Personalization in ECommerce (RPEC 2002) Malaga, Spain

[glance01] Glance, N. S. "Community Search Assistant", In Proceedings of IUI'01, Santa Fe, New Mexico, USA, January 2001

[goldberg92] Goldberg, D. Nichols, D. Oki, B. M. Terry, D. "Using Collaborative Filtering to Weave an Information Tapestry", Communications of the ACM, Vol. 35, No. 12, December 1992

[grasso99] Grasso, A. Koch, M. Rancati, A. "Augmenting Recommender Systems by Embedding Interfaces into Practices", In Proceedings of GROUP'99, Phoenix, Arizona, November 1999

[guarino95] Guarino, N., and Giaretta, P. (1995). "Ontologies and Knowledge bases: towards a terminological clarification", Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. N. Mars, IOS Press: 25-32

[harman86] Harman, D. "An Experimental Study of Factors Important in Document Ranking", Proceedings of 1986 ACM conference on Research and development in information retrieval, September 1986, Pisa Italy

[kautz97] Kautz, H. Selman, B. Shah, M. “Referral Web: Combining Social Networks and Collaborative Filtering”, Communications of the ACM 40(3), March 1997, 63-65

[kay90] Kay, A. “User interface: A personal view”, In: Laurel. B. (ed.). The art of Human-Computer Interface Design, Addison-Wesley, 1990, 191-207

[keogh99] Keogh, E. J. Pazzani, M. J. “Relevance Feedback Retrieval of Time Series Data”, In Proceedings of SIGIR '99, Aug 1999, Berkley, CA USA

[kobsa93] Kobsa, A. “User Modeling: Recent work, prospects and Hazards”, In Adaptive User Interfaces: Principles and Practice Schneider-Hufschmidt, M. Kühme, T. Malinowski, U. (ed) North-Holland 1993

[konstan97] Konstan, J. A. Miller, B. N. Maltz, D. Herlocker, J. L. Gordon, L. R. Riedl, J. “GroupLens: Applying Collaborative Filtering to Usenet News”, Communications of the ACM 40(3), March 1997, 77-87

[lang95] Lang, K. “NewsWeeder: Learning to Filter NetNews”, In ICML95 Conference Proceedings, 1995, 331-339

[larkey98] Larkey, L. S. “Automatic essay grading using text categorization techniques”, In Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval, Melbourne, AU, 1998

[lewis92] Lewis, D. D. “An evaluation of phrasal and clustered representations of a text categorization task”, In Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval, Kobenhaven, DK, 1992, 246-254

[linton99] Linton, F. “OWL - A Recommender System for IT Skills”, Workshop Interacting with Recommender Systems, CHI' 99, Pittsburgh, Pennsylvania, USA

[lotus01] Lotus, “Locating Organisational Expertise with the Lotus Discovery Server”, White Paper, 2001

[maes94] Maes, P. “Agents that reduce work and information overload”, Communications of the ACM 37(7) July 1994, 108-114

[maltz95] Maltz, D. Ehrlich, E. “Pointing the way: Active collaborative filtering”, CHI’95 Human Factors in Computing Systems, 1995

[mccallum00] McCallum, A. K. Nigam, K. Rennie, J. Seymore, K. “Automating the Construction of Internet Portals with Machine Learning”, Information Retrieval 3(2), 2000, pages 127-163

[mcdonald00] McDonald, D. W. Ackerman, M. S. “Expertise Recommender: A Flexible Recommendation System and Architecture”, In Proceedings of the ACM 2000 Conference on CSCW, Philadelphia, PA USA, December 2000

[melville02] Melville, P. Mooney, R. J. Nagarajan, R. “Content-Boosted Collaborative Filtering for Improved Recommendations”, In the Proceedings of the Eighteenth National Conference on Aritificial Intelligence (AAAI-2002), July 2002, Edmonton, Canada

[middleton01a] Middleton, S. E. “Interface agents: A review of the field”, Technical Report Number: ECSTR–IAM01-001, ISBN: 0854327320, University of Southampton, August 2001

[middleton02] Middleton, S. E. Alani, H. Shadbolt, N. R. De Roure, D.C. “Exploiting Synergy Between Ontologies and Recommender Systems”, International Workshop on the Semantic Web, Proceedings of the 11th International World Wide Web Conference WWW-2002, 2002, Hawaii, USA

[middleton01b] Middleton, S. E. De Roure, D. C. Shadbolt, N. R. "Capturing Knowledge of User Preferences: ontologies on recommender systems", In Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001), Oct 2001, Victoria, B.C. Canada

[mitchell97] Mitchell, T. M. “Machine Learning”, McGraw-Hill, 1997

[mladenić96] Mladenić, D. “Personal WebWatcher: design and implementation”, Technical Report IJS-DP-7472, Department for Intelligent Systems, J. Stefan Institute, 1996

[mladenić99] Mladenić, D. Stefan, J. “Text-Learning and Related Intelligent

Agents: A Survey", IEEE Intelligent Systems, 1999, 44-54

[mooney00] Mooney, R. J. Roy, L. "Content-Based Book Recommending Using Learning for Text Categorization", Proceedings of DL-00, 5th ACM Conference on Digital Libraries, 2000, San Antonio, US, ACM Press, New York, US

[porter80] Porter, M. "An algorithm for suffix stripping", Program 14 (3), July 1980, pp. 130-137

[rashid02] Rashid, A. M. Albert, I. Cosley, D. Lam, S. K. McNee, S. Konstan, J. A., Riedl, J. "Getting to Know You: Learning New User Preferences in Recommender Systems", In Proceedings of the 2002 International Conference on Intelligent User Interfaces, San Francisco, CA, pp. 127-134

[resnick97] Resnick, P. Varian, H. R. "Recommender systems", Communications of the ACM 40(3) March 1997, 56-58

[rich79] Rich, E. "User modelling via Stereotypes", Cognitive Science 3 1979, 329-354

[rucker97] Rucker, J. Polanco, M.J. "Siteseer: Personalized Navigation for the Web", Communications of the ACM 40(3), March 1997, 73-75

[schafer99] Schafer, J.B. Konstan, J. Riedl, J. "Recommender Systems in E-Commerce", In Proceedings of the ACM E-Commerce 1999 Conference, Denver, Colorado, 1999

[schwab00] Schwab, I. Pohl, W. Koychev, I. "Learning to Recommend from Positive Evidence", Proceedings of Intelligent User Interfaces 2000, ACM Press, pp 241-247

[sebastiani02] Sebastiani, F. "Machine learning in automated text categorization", ACM Computing Surveys, 2002

[shadbolt99] Shadbolt, N. O'Hara, K. Crow, L. "The experimental evaluation of knowledge acquisition techniques and methods: history, problems and new directions", International Journal of Human-Computer Studies (1999) 51, pp 729-755

[smart74] SMART Staff, "User's Manual for the SMART Information Retrieval System", Technical Report 71-95, Revised April 1974, Cornell University (1974)

[svensson01] Svensson, M. Höök, K. Laaksolahti, J. Waern, A. “Social Navigation of Food Recipes”, In Proceedings of SIGCHI’01, Seattle, WA, USA, April 2001

[sycara92] Sycara, K. Guttal, R. Koning, J. Narasimhan, S. Navinchandra, D. “CADET: A case-based synthesis tool for engineering design”, International Journal of Expert Systems, 4(2), 1992, 157-188

[tatemura00] Tatemura, J. “Virtual Reviewers for Collaborative Exploration of Movie Reviews”, In Proceedings of IUI’2000, New Orleans, LA, USA, 2000

[terveen97] Terveen, L. Hill, W. Amento, B. McDonald, D. Crester, J. “PHOAKS: A System for Sharing Recommendations”, Communications of the ACM 40(3), March 1997, 59-62

[van rijsbergen79] van Rijsbergen, C. J. “Information Retrieval (Second Edition)”, Butterworths, 1979

[voss97] Voss, A. Kreifelts, T. “SOAP: Social Agents Providing People with Useful Information”, Proceedings of the international ACM SIGGROUP conference on Supporting group work (GROUP'97), Phoenix AZ, 1997, pp 291-298

[wasfi99] Wasfi, A. M. A. “Collecting User Access Patterns for Building User Profiles and Collaborative Filtering”, In Proceedings of the 1999 International Conference on Intelligent User Interfaces, pages 57-64, 1999

[wenger99] Wenger, E. “Communities of practice: the key to knowledge strategy. Knowledge Directions”, The Journal of the Institute for Knowledge Management, 1, 48-93, 1999

[wenger00] Wenger, E. C., and Snyder, W. M. “Communities of Practice: The Organizational Frontier”, Harvard Business Review. January-February: 139-145. 2000

[witten00] Witten, I. H. Frank, E. “Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations”, 2000, Morgan Kaufmann publishers.

Index