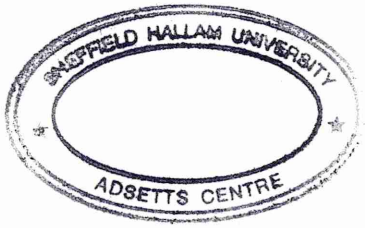
Evolutionary Learning Multi-Agent Based Information Retrieval Systems

Saeedeh Maleki-Dizaji

A thesis submitted in Partial fulfilment of the requirements for the Degree of Doctor of Philosophy at Sheffield Hallam University

October 2003



ABSTRACT

The volume and variety of information available on the Internet has experienced exponential growth, presenting a difficulty to users wishing to obtain information that accurately matches their interests. A number of factors affect the accuracy of matching user interests and the retrieved documents. First, is the fact that users often do not present queries to information retrieval systems in the form that optimally represents the information they want. Secondly, the measure of a document's relevance is highly subjective and variable between different users. This thesis addresses this problem with an adaptive approach that relies on evolutionary user-modelling. The proposed information retrieval system learns user needs from user-provided relevance feedback. The method combines a qualitative feedback measure obtained using fuzzy inference, and quantitative feedback based on evolutionary algorithms (Genetic Algorithms) fitness measures. Furthermore, the retrieval system's design approach is based on a multi-agent design approach, in order to handle the complexities of the information retrieval system including: document indexing, relevance feedback, user modelling, filtering and ranking the retrieve documents. The major contribution of this research are the combination of genetic algorithms and fuzzy relevance feedback for modelling adaptive behaviour, which is compared against conventional relevance feedback. Novel Genetic Algorithms operators are proposed within the context of textual; the encoding and vector space model for document representation is generalised within the same context.

ACKNOWLEDGEMENT

First of all I would like to sincerely thank my supervisor Dr. Henry Nyongesa, and Director of Studies Prof. Jawed Siddiqi, both of the School of Computing and Management Sciences, Sheffield Hallam University. They helped me and encouraged me to finalise the work described in this thesis, even if this, quite often, seemed impossible to do. Moreover, they provided me with many useful suggestions and remarks on writing this thesis.

I would also like to thank all my friends and colleagues in Computing Research Centre who have provided me with helpful suggestions, which have all been beneficial to this research, especially Amir Naghsh and Zulaiha Ali-Othman, who helped to test and evaluate my system.

Last but not least I would like to thank my husband Manuchehr, who has always been there when I needed moral support, and for his encouragement, help and tolerance during the last five years, and my children Pouyan and Arman

LIST OF PUBLICATIONS

S. Maleki-Dizaji, Z. AliOthman, H.O. Nyongesa and J. I. Siddiqi, **"Utilising a Multi-agent Approach for Designing An Adaptive Search Engine",** IEEE/WIC WI-2003, October 13-17, 2003 , Halifax, Canada.

H.O.Nyongesa, S. Maleki-Dizaji, Z. AliOthman, **An Adaptive Multi-Agent Based Search Engine,** EuroAsia – ICT 2002: Agents for Information Management, October 2002, Shiraz, Iran,

S. Maleki-Dizaji, H.O. Nyongesa, J. Siddiqi, **Fuzzy Relevance Feedback and Evolutionary Reinforcement in Adaptive Information Retrieval Systems",** CSI's 7th annual conference, 2002, Tehran, Iran.

H. O. Nyongesa, S. Maleki-Dizaji, J. Siddiqi, **Evolutionary User modelling in an Adaptive Information Retrieval System** , ICMLA'02 - June , 2002, Las Vegas, USA.

S. Maleki-Dizaji, H.O.Nyongesa, J.Siddiqi, **Adaptive multi-agent system for information retrieval,** Complex Adaptive Structures, 4-6 June 2001, Hutchinson Island, Florida USA.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1. INTRODUCTION

1.1. Motivation and Background

In recent years, we have witnessed a massive growth in the availability of on-line information. In particular, with the emergence of the World Wide Web (WWW), users now have access to millions of information sources, containing vast amount of documents. At the same time, there has also been a strong focus on digital libraries as a means of making such on-line information readily and easily available. Given that much of this information is textual in nature, the question that arises is how the access to this information can be efficiently facilitated. It is necessary to build tools aimed at helping users find those documents that satisfy their needs accurately.

The current methods for accessing large text collections (such as WWW) retrieve documents from a collection in response to a user's query. Such methods are best exemplified by search engines that are currently popular on the WWW, such as Yahoo, Google and Alta vista. In this paradigm, a user is required to specify his or her information need in the form of a *query* which is then compared (typically at a simple keyword level) with documents in a collection to find those likely to be most related to the query. Unfortunately, such approaches are quickly becoming inadequate, because their response to a user's query consists of a simple list of documents. Users then often find themselves having to wade through several hundred documents returned in response [Poulter 1997]. This is a time-consuming exercise.

An optimal Information Retrieval System (IRS) is one which is able to retrieve from a database only those documents that are *relevant* to a user's information needs, but excluding documents that are irrelevant. The concept of *relevance,* however, is one that is subjective and influenced by a variety of factors. First and foremost, the queries posed to information retrieval systems are, in most cases, not optimal in terms of describing the required information, with respect to an individual user's information needs. Among the factors that influence the *relevance* of a retrieved document are: user knowledge level, user perception, information currency and clarity. In spite of great improvements in the efficiency of search engines, many return a large proportion of

non-relevant documents. Hence recently, there has been a definitive paradigm shift from a view of relevance as simple term matching between query and document, to a view of relevance as a cognitive and dynamic process involving interaction between the information user and the information source.

This research addressed the problem of improving relevance of the documents returned by an information retrieval system. It is proposed that an important element to solving this problem is adding some "intelligence" to an IRS, based on ability of the IRS, to build a representation of user information needs (user models) instead of using the traditional keyword-based approach [Chau and Yeh 2000].

The overall goal of this research is to develop an IR system which uses the World Wide Web as a source of information. Any commercial search engine such as Google can be used to retrieve text from the Web. Google and many other search engines provide APIs that allow a programmer to interface with their content, and retrieve the data in a more convenient form. Thereafter we will process that data to find sets of relevant document. In this research, however, BIDS was used, since the proposed IRS is text based only.

Although keyword-based systems have been popular due to their simplicity, the performance of such systems ultimately suffers from what is known as the *keyword barrier*; the keyword barrier arises because keyword-based systems do not actually consider the semantic context of the input articles. There are two conventional approaches to developing user models in IRS: a knowledge-based approach [Chen and Dhar 1995] and a machine-learning approach [Chen and Shankaranarayanan 1998]. The knowledge-based approach involves endowing the IRS with a great deal of domain-specific background knowledge about different users, gleaned from human experts, to be able to create clusters or stereotypes of users. Such knowledge is then used to customise the dialogue with the user. The main disadvantages of this approach are that the system is only able to perform to the extent that it is programmed, and significant effort and *a priori* knowledge is required from the domain experts. In the machine learning approach by contrast, the IRS is designed to autonomously acquire knowledge through interactions with the user. The advantages of this approach are that it requires less effort from users and the system is able to adapt to changes in user-needs over time. The use of machine learning techniques for generating and adapting user models in information retrieval applications is therefore compelling. So far, however, the performance of systems built using both approaches is still relatively low.

In this thesis an adaptive IRS is proposed, which adopts the machine learning approach to provide a capability to learn the user-needs, in order to be able to filter documents to match individual needs. Reinforcement learning is used to adopt the most significant terms that best represent the user's interests. User preferences are explicitly acquired from user relevance feedback. Information preferences vary greatly across users; therefore, retrieval systems must be highly personalized to serve the individual interests of the user. A personalized retrieval system must satisfy three functional requirements:

Specialization: A personalized retrieval system must serve the special interests of the user. The system selects articles deemed to be interesting to the user and eliminates the rest. However, an adaptive system should be able to effectively differentiate the articles that are actually relevant to the user from similar ones that are not. Effectiveness means the proportion of irrelevant articles delivered to the user should be as low as possible, while the proportion of relevant articles eliminated should also be low.

Adaptation: Adaptation involves repeated interactions with the user, in order to identify patterns in the users' behaviour. The retrieval system must infer user habits and specialize to them, and thus, recommend as many relevant articles and as few irrelevant articles as possible. Since retrieval typically involves interaction over long periods of time, user interests cannot be assumed to stay constant. When they change, the system must be able to notice that the user's interests have changed, and adapt its behaviour in response to the change. Anticipating and adapting to user needs also helps make the system more user-friendly. This is essential if more and more people are to use the system.

Exploration: An adaptive system should also be capable of exploring newer information domains to suggest items of potential interest to the user. There are two motivations for exploration. One is that exploration helps match a presently unknown user interest. The other motivation is that it helps improve the adaptation process. This is also necessary because newer kinds of information need to be explored to serve changing user interests.

The above requirements, in combination with the other common information retrieval features, such as, document representation and ranking, lead to a system of relatively high complexity, because various times consuming tasks need to be carried out. It is

proposed that a solution to this problem is a multi-agent approach. A multi-agent approach would make the system more flexible and reliable and, also promises the following non-functional requirements:

- Faster access times: utilising parallel task execution using agents increases the number of retrieved documents.
- Robustness: Agents operate independent of each other; hence, the system can retain some functionality when one or more agents fail.
- Ease of maintenance: each agent is designed for a specific task, and maintenance of each agent can be done without affecting other agents.

1.2. Contributions

This thesis will present novel use of a combination of three powerful machine learning techniques for modelling adaptive behaviour in adaptive information retrieval system, which not only learns to retrieve relevant information, but also continuously adapts to improve the quality of future search results. The approach aims to increase retrieval precision, with minimum penalty on recall, by adapting to new information needs within a specific subject area over a period of time. This is achieved by applying fuzzy relevance feedback and evolutionary learning, with genetic algorithms (GA). The thesis makes the following broad contributions to the field of information retrieval to satisfy functional and non-functional requirements:

- The use of Genetic Algorithms for user-modelling of adaptive and exploratory behaviour in information retrieval systems as proposed in our new machine learning approach discussed above is validated and demonstrates that in utilising this approach, more documents *relevant* to a user's query can be retrieved in comparison with traditional relevance feedback method (section 5.3.3.4).
- The proposed adaptive IRS satisfies the functional requirements of exploratory behaviour. That is, exploring newer information domains and suggest items of potential interest to the user. Therefore, novel crossover and mutation GA operators together with a new method for chromosome encoding have been implemented to demonstrate exploratory behaviour of functional requirements (section 5.3.3.4).

- To achieve the functional requirements for specialization behaviour, the proposed system utilises a method to rank the retrieved documents efficiently. That is, the vector space model for document representation is generalized for use with documents containing textual information. The distance metric for computation of document similarities and the effect of relevance feedback have also been generalized (section 5.3.3.3).
- Develop and evaluate an IRS using a multi-agent system approach to simplify design and development of complex systems that are more robust and easier to maintain and satisfies non-functional requirement (section 5.2 and 5.3.4).
- To simplify design and development of complex systems that are more robust and easier to maintain, an IRS has been developed and evaluated utilising a multi-agent system approach that satisfies non-functional requirements as noted in sections 5.2 and 5.3.4.
- To measure IRS retrieval performance and determine precision and recall, new measurements have been proposed utilising fuzzy feedback (section 7.3.3).

1.3. Thesis Outline

In chapter two we formulated the personalisation of information retrieval as a problem of user modelling in order for it to be adaptive. We surveyed the literature of machine learning techniques specially: fuzzy logic, Genetic Algorithms, relevance feedback and proposed that combining their strengths would be a novel approach to adaptively.

In chapter three we briefly compare information retrieval (IR) and information filtering (IF) more significantly we proposed that adaptive user modelling could be achieved via an adaptive system for information retrieval. We therefore surveyed the literature in IR to capitalise on Information Retrieval models, and the advantages and disadvantages of each model. This Chapter also compares and contrasts Information Retrieval (IR) and Information Filtering (IF) techniques.

In the previous two chapters having formulated the problem and defined the conceptual solution, in chapter four we presented a survey of relevant aspect of agent systems to show how in particular a multi-agent is appropriate paradigm to be used to design and implement systems.

In chapter five we present the complete system development life cycle. It makes explicit adaptation of techniques that we have carried out to achieve a system that satisfies the function and non-functional requirements in this chapter.

In chapter six discusses the process of choosing a suitable agent framework to deploy the proposed system in chapter five. We present a survey that includes a comparison of various agent development environments.

Chapter seven discusses an evaluation of the proposed adaptive IRS. The set of experimental results presented in this Chapter include the comparison of the effectiveness of the evolutionary adaptive information retrieval system and traditional relevance feedback information retrieval system .

In chapter eight discusses the impact of this research, identifies limitations and drawbacks of the system and discusses the possible areas of research for future work.

Chapter 2. ADAPTIVE USER MODELLING TECHNIQUES

2.1. Introduction

Large quantities of Information are now becoming available in the form of text, audio, video, or image repositories. Personalizing the access to the information is a challenging application for research on user modelling and adaptively. An optimal Information Retrieval System (IRS) should be able to retrieve from a database, only those documents that are *relevant* to a user's query. In practice, unfortunately, most IR systems return many non-relevant documents. We have proposed that this issue can be dealt with by adding some intelligence and personalization to an IRS, in order that its performance can improve over time. In order to achieve this, it is desirable adaptive systems which can give a solution for two IRS problems, different people are different and individuals are different at different times.

User adaptive system can be worthwhile for a system to learn something about each individual user and adapt its behaviour to them in some nontrivial way. However it is system that offers their user the capability to select, or set between different alternative presentation and interaction characteristic, among the ones built into the system figure 2-1.

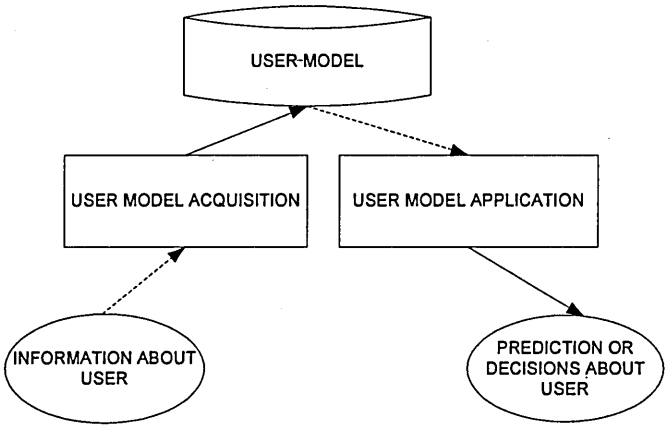


Figure 2-1: General schema for the processing in user-adaptive system.

(Oval: input or output; Rectangles: processing method; Cylinder: stored information. Dotted arrows: use of information; solid arrows: production of results.)

Adaptive systems monitor the user's activity pattern and automatically adjust the interface or content provided by the system to accommodate such user differences as well as changes in user skills, knowledge and preferences. *Adaptable* systems allow the user to control these adjustments, often providing guidance or specialized help to the user. Figure 2-2 shows general design model of an adaptive IRS.

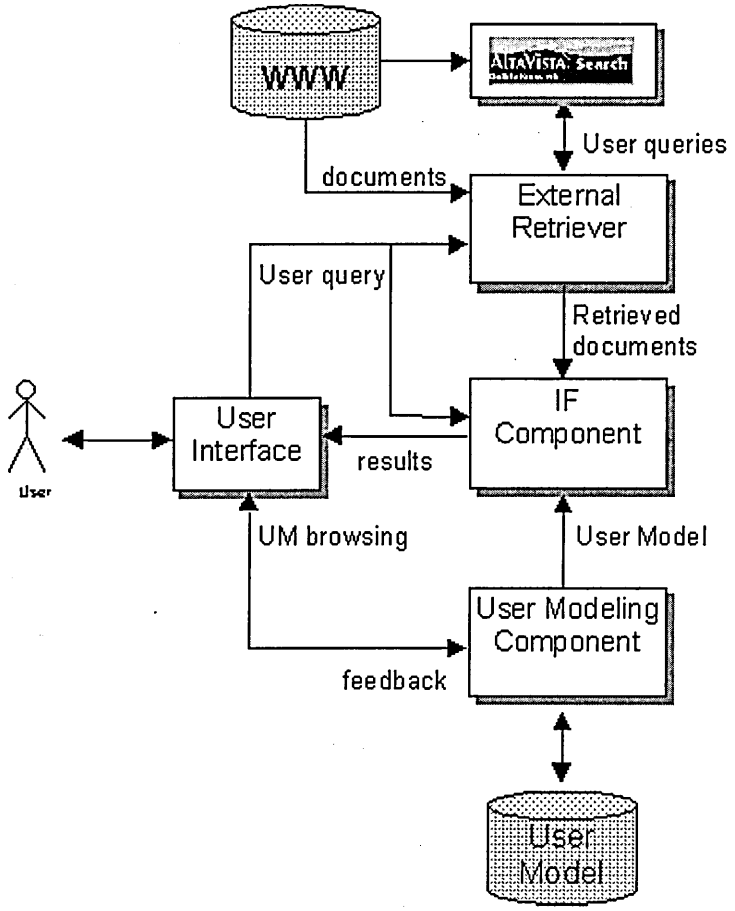


Figure 2-2: Adaptive Information Retrieval System

If a program can change its behaviour based on something related to the user, then the program does (implicit or explicit) user modelling.

The user model contains all information that the system knows about the user. It is generally initialized either with default values or by querying the user. Thereafter, it is maintained by the system, although the user may be able to review and edit their profile. User actions and events at various conceptual levels, such as mouse clicks, task

completion and requests for help, are reported by the user interface or core application to the user profile.

2.2. User Modelling

User modelling can be defined as the effort to create a profile of the user's interests and habits and employ the profile in order to improve the efficiency of the system [Minio 1996]. The idea behind user modeling is as follows: if the system can create an accurate representation of the user, it can anticipate and predict the user's actions, and needs with greater reliability. Therefore, it makes the interaction process more effective. Typical attributes maintained in the user model [Kosba 2001]:

- User preferences, interests, attitudes and goals
- Proficiencies (e.g. task domain knowledge, proficiency with system)
- Interaction history (e.g., interface features used, tasks performed/in progress, goals attempted/achieved, number of requests for help)
- User classification (stereotype)

User modelling techniques differ in the ways they acquire, use and represent a user profile. Profiles can be acquired or generated in a variety of ways:

- By direct user interviews.
- By "knowledge engineers" using user stereotypes. *User Clustering* creates groups according to interests that are shared by users. For instance, the stereotype of "Computer Science" would include a sub-category "Programming" into the profile.
- Machine learning techniques like inference, induction, where the modeller tries to identify certain patterns in the user's behaviour.
- Profile building by example, where the user provides examples of his/her interest.
- Rule-based profiles, where the users specify their own rules in the profile, rules that control the behaviour of the model under certain trigger conditions.

The above methods have their advantages and disadvantages, but in general the most successful are those that try to analyse the information not just at a keyword level, but

also at contextual and semantic level. User profiles can be represented using a wide range of techniques from simple keyword-based files to artificial-intelligence based representations. The representational formats include keyword-base profiles, rule-based profiles, vector space presentation and neural network-based representations. It is useful to note that a typical user has multiple and sometimes overlapping interests. The categories of interests in stereotype user profiles must be very fine-grained and users have to select categories individually in order to build their profiles. User interviews are very time-consuming, and sometimes users fail to identify properly, categories of interest. Moreover, the cost of maintenance of such models is very high.

One approach for constructing a user profile is to describe the profile through a set of keywords and to require the user to provide the necessary keywords. The approach however, is simplistic and requires much effort from the user. In fact, if the user is not familiar with the service which generates the upcoming documents, he/she might find it fairly difficult to provide the keywords which appropriately describe his/her preferences in that context. Furthermore, attempts by users to familiarise themselves with the vocabulary of the upcoming documents might turn into a tedious and time consuming exercise. Thus, in spite of its feasibility, requiring users to describe their profiles precisely might be impractical. A more elaborate alternative is to collect information from users about their preferences and to use this information to build user profiles dynamically. This can be accomplished as follows. In the very beginning the user provides a set of keywords which describe an initial (and primitive) profile of preferences. As new documents arrive, the system uses this profile to select documents which are potentially of interest and shows them to the user. The user then goes through a feedback cycle in which he/she indicates not only the documents which are really relevant but also the documents which are non-relevant [Salton and Buckley 1997b]. The system uses this information to adjust the user's profile description such that it reflects the new preferences just declared. Of course, with this procedure, the profile is continually changing. Hopefully, however, it stabilises after a while and no longer changes drastically (unless, of course, the user's interests shift suddenly).

Traditional user modelling systems often make use of knowledge representation techniques. KR formalisms offer facilities for maintaining knowledge bases (using representation formalisms) and for reasoning (using the inference procedures of representation formalisms). For user modelling, these facilities are typically employed

as follows: Assumptions about individual characteristics of the user are maintained in a knowledge base, using representation formalism. Since this knowledge base may additionally contain system knowledge about the application domain or meta-knowledge for inferring additional assumptions about the user from her current model (including stereotypes), it has been called user modelling knowledge base. If available, inference procedures of the representation formalism or meta-level inferences can be used to expand the user model. The use of (particularly logic-based) knowledge representation methods in user modelling systems has been analysed in detail by Kosba (2001).

The knowledge-based approach involves endowing the IRS with a great deal of domain-specific background knowledge about the user, in order to be able to create clusters or stereotypes. In such a system, knowledge about the user is used to customize the dialogue with the user. Examples of this include, Desire [Treur et al., 1999] and Profusion Personal Assistant [Edgar 1998, Fan and Gauch 1997]. The main disadvantages of the knowledge-based approach are, that systems are only able to carry out what they were programmed to do, and a significant effort and knowledge are required from domain experts.

In a machine learning (ML) approach, the system autonomously attempts to acquire knowledge through inference from user behaviour. The advantages of this approach are, it requires less effort from users and, the system is able to adapt to changes in user behaviour. Several techniques have been proposed for this purpose, including relevance feedback (RF) [Salton and Buckley 1998], symbolic learning techniques, such as decision trees [Chen and Dhar 1995], Genetic Algorithms (GA) [Vrajitoru 1998], simulated annealing [Chen and Shankaranarayanan 1998, Yang 2000] and fuzzy logic [Srinivasan et al., 2001].

2.3. User Modelling: A Review

Adaptive systems have been designed or implemented for a variety of purposes. The following six systems illustrate aspects of the user model described above.

P-TIMS [UM97] -- A commercial financial management system implements to add an adaptive and adaptable interface using a simple user model and rule set. As the user

spends more time using the system and uses more complex functions, the system reveals a more extensive interface. The user model is explicitly exposed by providing a "preferences" dialog box, which the user can adjust at any time. Empirical evaluation showed that it improved a subjective measure of user satisfaction.

AVANTI [Fink et al 1996, 1997] -- A hypermedia information system about a metropolitan area (e.g. public services, transportation, buildings) for a variety of users with different needs (e.g., tourists, residents, elderly people, blind persons, wheelchair-bound people and users with slight forms of dystrophy). It uses an initial interview to create the initial primary assumptions (i.e., user profile), draws inferences to generate additional assumptions, and uses stereotypes for certain subgroups of users (e.g. tourists, blind users). It then customizes the web pages presented to the user. For example, wheelchair users are presented with information about physical accessibility of facilities when viewing relevant web pages.

Interbook [Bruislovsky and Schwarz 1997] -- An advanced WWW application that supports incremental learning and incremental interfaces. The web presents challenges when maintaining the user model. Adaptive systems derive much of their data for the user model from the user interface component, which can track user actions and report them in detail to the application user model. In a web environment, however, the amount and kind of detail about user actions is limited by the HTTP protocol and the HTML model built on it. Interbook addresses these problems by tracking what the users have seen, rather than what they have done, and using that to infer what the users know.

ORIMUHS [Encarnação and Miguel 1997] -- An adaptive hypermedia help system that supports context-sensitive and user-adaptive presentation of hypermedia help, providing user-controlled help adaptation and agent-based retrieval of additional information. It incorporates a sophisticated user model with stereotypes (levels of expertise), as well as agent-based retrieval of help information. ORIMUHS has been implemented in medical imaging and CAD systems.

Lumière [Horvitz 1998] -- Uses a Bayesian user model to infer a user's needs based on the user's background, actions and queries. Lumière prototypes served as the basis for the *Office Assistant* in Microsoft Office '97, which provides guidance and tips to the user.

2.4. Machine learning for user modelling

As part of the advent of the World Wide Web and the recent resurgence of machine learning for user modelling research, IR tasks have received much attention. This problem is well illustrated by the demands of user modelling for information retrieval. The main objective is to learn a model of the user's interests or information need, in order to facilitate retrieval of relevant information. Most work on content-based information filtering casts the automated acquisition of user profiles as a text classification task (for example, Pazzani and Billsus, 1997; Lang, 1995; Mooney and Roy, 1998). In these systems, a set of text documents rated by the user (e.g. interesting vs. not interesting) is used as the input for a learning algorithm, and the resulting classifier can be interpreted as an automatically induced model of the user's interests. An underlying assumption often made is that more training data leads to improved predictive performance.

However, if we take into account that a user's interests are dynamic and are likely to change over time, this assumption does not hold. A classifier built from a large number of training documents that accurately reflect the user's past interests is of limited practical use and might perform substantially worse than a classifier limited to recent data that reflects the user's current interests. This example illustrates that a good text classification algorithm is not necessarily a useful user modelling algorithm.

User model acquisition is a difficult problem. In Machine Learning, the information available to a user modelling system is usually limited, and it is hard to infer assumptions about the user that are strong enough to justify non-trivial conclusions. Classical acquisition methods like user interviews, application-specific heuristics, and stereotypical inferences often are inflexible and unsatisfying. In Information Retrieval, user models have been limited to lists of terms relevant to an information need. The list is usually very short for ad hoc querying and longer for information filtering tasks.

Information systems that could benefit from having a user model should be able to adapt to individual users, to learn about their preferences and attitudes during the interaction (to construct a user profile), and memorize them for later use. Moreover, these user profiles could represent a starting point for the creation of user communities

based on shared interests or goals. Further, the system should be able to update its model is a user changes interests.

IRS that learn from user feedback attempt to modify either the query representation *(query formulation or query refinement)* or the document representation *(document-vector modification)* [Sebastiani 2002]. Document-vector modification changes and improves document indexes based on the user feedback about relevant and irrelevant documents. Using such a technique, the vectors of documents previously retrieved in response to a given query are modified by moving relevant documents closer to the query and at the same time moving irrelevant documents away from the query. When a document is judged relevant to a query, the learning algorithm modifies the document representation for each feature present either in the query or in the document. When a document feature is present in the query (in a Vector Space Model, the set of features is the same for both queries and documents), its weight is increased, while, conversely, a when feature is not in the query, its weight is decreased (unless it was not present in the document representation).

Another use of machine learning techniques in IRS is generating and maintaining user profiles by query refinement. In such a system, it is important to provide the user with interactive query refinement mechanisms. This can be done by the user providing feedback on retrieved documents as relevant or irrelevant. Based on this feedback, the system automatically refines the query and resubmits it to retrieve a new set of documents. The process continues until the system cannot provide any new documents that satisfy the latest version of the query.

In such a system user must perform additional work to provide explicit feedback to the system (by clicking on a button) but is not provided with an immediate reward. Users rarely provide information to the modelling system if they must go out of their way or if they see no immediate benefit.

One approach to this problem is to infer the labels from the user's behaviour. For example, the Letizia system (Lieberman, 1995) infers that a user is interested in a web page if a variety of actions are performed (e.g. printing the page or creating a bookmark), while the user is not interested under other circumstances (e.g. by quickly hitting the back button). Such implicit feedback methods allow a large amount of data to

be collected unobtrusively. One can imagine future systems that would use the user's facial expression, body language or other forms of implicit feedback for this purpose.

Another approach to the problem is to use a small initial body of labelled examples to infer labels for a larger body of examples which is then used to train the learning algorithm. This technique is related to the information retrieval method of pseudo-feedback (Kwok and Chan, 1998) in which first the system ends documents similar to the user's query and then it ends documents similar to the retrieved documents. However, in the machine learning approach (Nigam et al.,1998), the process of inferring the label for unseen documents is repeated until a stable solution is found via a procedure known as expectation maximization.

In this research, ML has been used for query refinement and user modelling to improve the quality of future search results. This involves a combination of using relevance feedback, GA and fuzzy inference. In the following sections briefly describes the three learning algorithms that I will be using throughout this thesis.

2.4.1. Relevance Feedback

Relevance feedback (RF) is a technique that allows a user to interactively express an information requirement by modifying successive query formulations using extraneous information, and is used for mainly text and image retrieval. This additional information is often provided by indicating some relevant documents among the documents retrieved by the system. RF is a good technique for specifying an information need, because it releases the user from the burden of having to think up lots of terms for the query. Instead the user deals with the ideas and concepts contained in the documents. It also fits in well with the known human trait: "I don't know what I want, but I'll know it when I see it".

RF is a controlled automatic process for query reformulation. The objective is to perform a query alteration to move the query in the direction of relevant items and away from non-relevant ones [Salton and Buckley 1998]. User relevance feedback can be seen as an interactive process, where users are encouraged to apply domain knowledge to generate more comprehensive queries. The main assumption behind relevance feedback is that documents relevant to a particular query resemble each other in the sense that it is represented by a similar vector of keywords or descriptors [Salton and

Buckley 1998]. A user indicates to the system which of the documents from the collection just retrieved are considered to be relevant to the query. The query formulation can then be improved by increasing its similarity to such previously retrieved relevant documents. There are two stages in the application of this method: Firstly, query expansion is used to select a set of terms of the relevant documents and add these terms to user queries, or remove the terms appearing in the non-relevant documents. Secondly in query re-weighting terms appearing in relevant documents are given increased term weights, and conversely, terms in non-relevant documents have decreased term weights. The query is modified according to the following equation [Salton and Buckley 1998]:

$$Q' = Q + \frac{1}{n1}\sum_{i=1}^{n1}(d_i) - \frac{1}{n2}\sum_{i=1}^{n2}(d_i') \qquad (2\text{-}1)$$

Where

Q' = is the vector for the new query,

Q = is the vector for the initial query,

d_i = the vector for relevant documents,

d'_i = the vector for non-relevant documents,

$n1$ = the number of relevant documents,

$n2$ = the number of non-relevant documents.

Probabilistic relevance feedback (PRF) is one of the most advanced RF techniques applied in IRS. Briefly, the technique consists of adding new terms to the original query. The terms added are chosen by taking the first m terms in a list where all the terms present in relevant documents are ranked according to their weight. Seo and Zhang [2000] proposed an information filtering agent to learn users' preferences, using reinforcement learning to adapt the most significant terms that best represent user's interests. In contrast to conventional relevance feedback methods, which require explicit user feedback, this approach learns user preferences implicitly from observations of browsing behaviours during interaction. Neural relevance feedback (NRF) is another approach for relevance feedback that uses neural networks. NRF learns using training examples to associate new terms to the original query formulation. While NRF acts in a

similar way to classical RF, the main difference is that the weights used to order and select the terms are obtained from the output of a 3–layer feed-forward NN trained using the Back Propagation (BP) learning algorithm. Each node in the NN input layer represents a query term and each node in the output layer represent a document term [Crestani 1995].

2.4.2. Genetic Algorithms

The GA is an optimisation algorithm, mimicking Darwinian evolution which was introduced by Holland [1992].

GA is especially well suited to problems which cannot be solved by mathematical evaluation or classical gradient descent strategies, and for which exhaustive search of the solution space is computationally infeasible. In GA, there is a set of candidate solutions to a problem; typically, this set is initially filled with random trial solutions. A candidate solution to a problem is called a *chromosome.* Each candidate is typically an ordered fixed length (not in all GAs) array of values for attributes ('genes'). Thus, in building a GA for a specific problem, the first task is to decide how to represent possible solutions. After deciding a representation, a GA usually proceeds in the following steps [Holland 1992]:

A. *Initialization*: A set of candidate solutions is randomly generated.

B. *Iterate through the following steps*, until some termination criterion is met (such as no improvement in the best solution so far after some specified time, or until a solution has been found whose fitness is better than a given 'adequate' value).

   1. *Evaluation:* Using some predefined problem-specific measure of fitness that evaluates each candidate (chromosome) as to how good a solution to the problem it is. The measure is called the candidate's fitness

   2. *Selection:* Select pairs of chromosome from current generation to be used for recombination. This may be done entirely randomly, or based on fitness.

   3. *GA operators:* Produce new individuals by using genetic operators on the individuals chosen in the selection step. The purpose of such operators is to

enable the evolutionary process to move towards new ``promising" regions of the search space. There are two main kinds of operators:

Crossover: A new individual is produced by recombining features of a pair of *'parent'* solutions.

Mutation: A new individual is produced by slightly changing an existing one.

The performance of GA very much depends on these operators, while the implementation depends on the encoding method, and also on the problem domain. Crossover is performed between two selected individuals, called *parents*, by exchanging parts of their genomes to form two new individuals, called *offspring*. In its simplest form, sub-strings are exchanged at a randomly selected crossover point. They can produce an individual better than both parents; if the offspring is poor it will have a lower chance of selection later on. In any event, features of the parents appear in different combinations in the offspring. Mutation, on the other hand, serves to allow local hill-climbing, as well as introduce variation which cannot be introduced by recombination.

4. *Population update:* the population is changed by choosing to remove some or all of the individuals in the existing generation (usually beginning with the least fit) and replacing these with individuals produced by GAs operators. The new population thus produced becomes the current generation. Figure 2-3 presents the standard genetic algorithm in pseudo-code format.

```
Begin GA
   g:=0  { generation counter }
   Initialize population P(g)
   Evaluate population P(g)  { i.e., compute fitness values }
   while (not  termination condition)  do
      g:=g+1
      Select P(g) from P(g-1)
      Crossover P(g)
      Mutate P(g)
      Evaluate P(g)
   end while
end GA
```

Figure 2-3: Pseudo-code of the standard genetic algorithm.

Representation (encoding)

A GA is an iterative procedure that consists of a constant-size population of individuals, each one represented by a finite string of symbols, known as the *genome*, encoding a possible solution in a given problem space. The symbol alphabet used is often binary, though other representations have also been used, including permutation encoding, value encoding and tree encoding [Koza 1992].

Selection

To form a new population (the next generation), individuals are *selected* probabilistically, but in relation to their fitness. Many selection procedures are currently in use, one of the simplest being *fitness-proportionate selection* [Holland 1992], where individuals are selected with a probability proportional to their relative fitness. This ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population. Thus, high-fitness "good " individuals stand a better chance of "reproducing", while low-fitness ones are more likely to disappear, this is what is known as *survival of the fittest.* A common method for selection is *roulette wheel.*

Given a population size of p, the relative fitness of each chromosome f_i the probability of selecting i^{th} chromosome x_i is given by:

$$P(X = x_i) = \frac{f_i}{\sum_{i=1}^{p} f_i} \tag{2-2}$$

The higher the fitness a chromosome has, the more chances it has to be selected. A roulette wheel on which are placed all chromosomes in the population, each represented by a sector according to its fitness (Figure 2-4). Chromosomes with larger fitness will be selected more times.

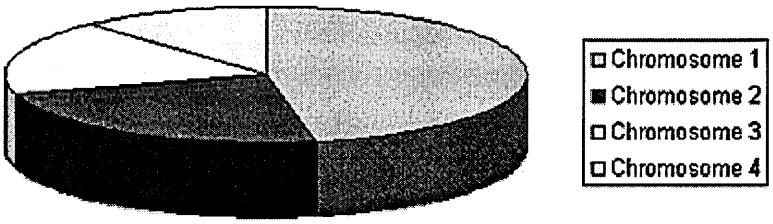


Figure 2-4: Example of roulette wheel method

Crossover

The crossover operator is the most important operator in GAs. Crossover is performed between two selected individuals, called *parents*, by exchanging parts of their genomes to form two new individuals, called *offspring*. In its simplest form, substrings are exchanged at a randomly selected crossover point. This operator tends to enable the evolutionary process to move towards new "promising" regions of the search space. Crossover is performed with probability p_c *(*called the *crossover rate)*. The crossover rate p_c is a number between 0 and 1, determining the probability that this exchange of genetic information actually happens for the pair of parents. In practice, a rate somewhere between 0.6 and 0.9 gives the best results. There are several forms of crossover such as, one-point crossover, two-point crossover, n-point crossover, uniform crossover and fusion crossover [Goldberg 1989, Koza 1992].

Mutation

Mutation is important to maintain diversity within the population. Crossover alone is not able to introduce a new value at a certain bit position when all individuals in the

population have the same value at this bit position. So, the task of mutation is primarily to prevent the population from converging at one specific point, and continue to maintain some evolvability. It is carried out by flipping bits at random, with a small probability p_m *(mutation rate)*. Mutation rate determines how often parts of chromosome will be altered. If there is no mutation, offspring are taken after crossover (or copy) without any change. If mutation is performed, part of chromosome will be changed. In practice, the mutation rate will be very small, in the order of 0.01 to 0.001. Two examples of mutation operation are:

Swap: Randomly select two positions and swap the elements in those positions.

Shift: Randomly select an element and put that element in a random position, shifting the other elements right. An alternative form is to select a random subsequence and insert it in a random position.

2.4.3. Fuzzy Logic System

Fuzzy set theory is a mathematical concept proposed by Zadeh in [Zadeh 1965]. This concept helps to improve the relationship between human and computers. In this section we will give a review of the basic ideas of fuzzy set theory needed to support fuzzy inference systems (FIS).

Fuzzy logic may be viewed as a methodology for computing with words that consists of two elements of functions and rules described in linguistic terms, rather than numbers. Although words are less precise than numbers their use is closer to human perception [Zadeh 1984]. Fuzzy logic has been applied virtually in all branches of science, engineering and economics. Fuzzy logic, unlike classical set logic, which assigns "crisp" values of true or false to statements, allows for a continuous range of truth values on the interval [0,1]. Fuzzy logic overcomes the principle of the excluded middle, where it is stated that a statement is either true or it is false. Instead, a statement in fuzzy logic is true to some degree, represented by a number between 0 and 1, and, at the same time, it is false to some degree. This principle allows for the development of fuzzy set theory, which according to its advocates corresponds to the everyday reality much more closely than conventional set theory. A conventional set thus becomes a special case of a fuzzy set in which the degree of membership for each element can only be 0 or 1. One way to look at a fuzzy set is to view it as a set of pairs, where the first

element of each pair is the actual element of the set, and the second is its degree of membership. Also, one can consider the mapping between the elements and the membership values to be a function (usually referred to as the membership function), which completely defines the fuzzy set. Fuzzy sets turn out to be very useful, because they allow us to represent mathematically everyday concepts that are often vague and uncertain, including representation of subjective relevance.

Fuzzy Set Theoretic Operations

The same operations which can be applied to crisp sets, are also applied to fuzzy sets, since crisp sets are a subset of fuzzy sets. Let A and B be two fuzzy sets in U with membership function μ_A and μ_B, respectively. The set operations of union, intersection, complement and other relations of fuzzy sets are defined by their membership function as follows:

Union: The membership function $\mu_{A\cup B}$ of the union $A\cup B$ is point-wise defined for all $u \in U$ by

$$\mu_{A\cup B}(u) = \max\{\mu_A(u), \mu_B(u)\} \qquad (2-3)$$

Intersection: The membership function $\mu_{A\cap B}$ of the intersection $A\cap B$ is pointwise defined for all $u \in U$ by

$$\mu_{A\cap B}(u) = \min\{\mu_A(u), \mu_B(u)\} \qquad (2-4)$$

Complement: The membership function of the complement of a fuzzy set A is point-wise defined for all $u \in U$ by

$$\mu_{\bar{A}}(u) = 1 - \mu_A(u) \qquad (2-5)$$

Cartesian Product: If $A_1,\ldots, A_n$ are fuzzy sets in $U_1,\ldots, U_n$, respectively, the Cartesian product of $A_1,\ldots, A_n$ is a fuzzy set in the product space $U_1 x \ldots x U_n$ with the membership function

$$\mu_{A_1 x..xA_n}(u_1,u_2,...,u_n) = \min\{\mu_{A_1}(u_1),...,\mu_{A_n}(u_n)\}$$

$$\mu_{A_1 x..xA_n}(u_1,u_2,...,u_n) = \mu_{A_1}(u_1).\mu_{A_2}(u_2)...\mu_{A_n}(u_n)$$

Or (2-6)

Fuzzy Relation: An n-array fuzzy relation is a fuzzy set in $U_1 x \ldots x U_n$ and is expressed as

$$R_{U_1 x..xU_n} = \{((u_1,...,u_n),\mu_R(u_1,...,u_n)) \mid (u_1,...,u_n) \in U_1 \times ... \times U_n\} \quad (2-7)$$

Sup-Star Composition: If R and S are fuzzy relations in UxV and VxW, respectively, the composition of R and S is a fuzzy relation denoted by R o S and is defined by

$$R \circ S = \{[(u,w), \sup_v(\mu_R(u,v) * \mu_S(v,w))], u \in U, v \in V, w \in W\} \quad (2-8)$$

Here * could be any operator in the class of triangular terms, namely, minimum, algebraic product, bounded product, or drastic product [Zadeh 1984].

Fuzzy Number: A fuzzy number F in a continuous universe U, e.g., a real line, is a fuzzy set F which is normal and convex, i.e. the use of fuzzy set provides a basis for systematic ways of content manipulation.

$$\begin{aligned} &\max_{u \in U} \mu_F(u) = 1; (normal) \\ &\mu_F(\lambda u_1 + (1-\lambda)u_2) \geq \min(\mu_F(u_1), \mu_F(u_2)); (convex) \\ &: u_1, u_2 \in U, \lambda \in [0,1] \end{aligned} \quad (2-9)$$

In particular, fuzzy sets can represent linguistic variables. A linguistic variable can be regarded here as a variable, the value of which is a fuzzy number, or as a variable, the values of which are defined in linguistic terms. A linguistic variable is characterized by a quintuple (x,T(x),U) in which x is the name of the variable. T(x) is the term set of x that is the set of linguistic values of x with each value being a fuzzy number defined in U. For example, if Temperature is interpreted as a linguistic variable, then its term set T(Temperature) could be T(Temperature) = { Cool, Warm, Hot }

Each term in T (Temperature) is characterized by a fuzzy set in the universe of discourse U=[0,40]. These terms can be characterized as fuzzy sets. Their membership functions are shown in Fig 2-5.

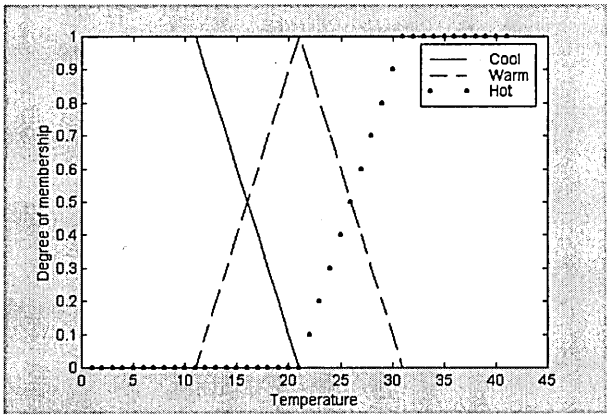


Figure 2-5: Diagrammatic representation of fuzzy temperatures with three sets.

Fuzzy Inference Systems

Fuzzy Inference Systems (FIS) are the most common practical use of fuzzy logic. They can be applied to virtually any domain. On the surface, fuzzy systems look like conventional rule-based systems. They are defined in terms of traditional "if-then" rules, such as If the temperature is "hot", then the speed of the motor (in the air conditioner) is "high". If the temperature is "cool", then the speed is "low". The difference, however, is the fact that "hot", "cool", "high", and "low" are represented as fuzzy sets. Because of that, for a statement such as "The temperature is hot", the value need not to be 0 or 1, but may be any number in between. Furthermore, in a conventional rule-based system, only one rule can "fire", or be active at any given time. In a fuzzy system, all rules fire simultaneously to various degrees. For example, a particular temperature value may belong to both "hot" and "cool" sets to a degree greater than zero, depending on how we define their membership functions. In this case both rules will fire. Essentially, any

inference system is a mapping between a set of inputs and a set of outputs. What makes a fuzzy system fundamentally different from any other inference system is the fact that a fuzzy system can be built using imprecise or incomplete knowledge, whereas to build a conventional inference system one needs to know the precise relationship between the inputs and outputs ahead of time. The inner workings of a fuzzy system can be broken down into three distinct steps (Figure 2-6).

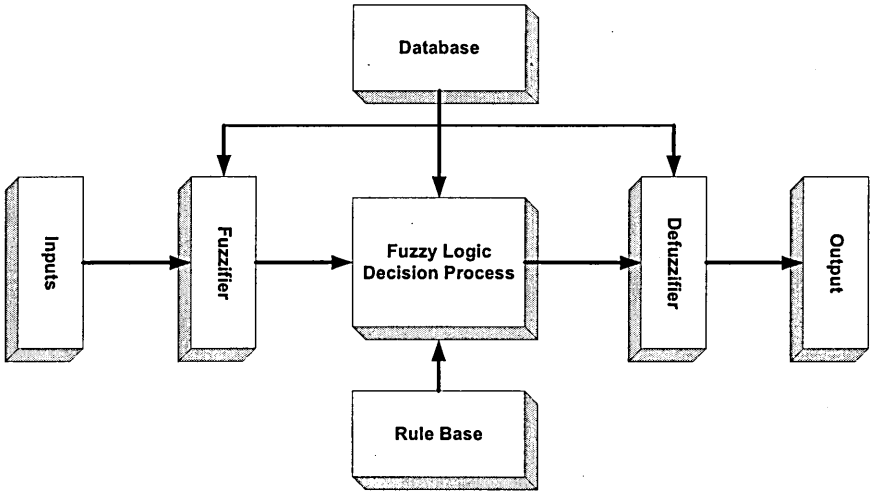


Figure 2-6: Fuzzy System Structure

The fuzzy system (Figure 2-6) steps are described in the following paragraphs.

Fuzzification

The first step is usually referred to as fuzzification. In this step we transform the crisp input values by determining their respective memberships in the input fuzzy sets. These memberships in turn determine the degrees to which each rule will fire.

There are various types of fuzzifiers that can be used such as Singleton, Gaussian or Trianglar fuzzifiers. In this work, triangular fuzzifiers are used [Kosko 1994].

Fuzzy Inference

After the inputs have been decomposed into fuzzy sets, they are applied to the set of rules that governs the system's behaviour for each combination of inputs. Each rule consists of conditions and actions where the condition is interpreted from input fuzzy set and the output is determined from the output fuzzy set.

Rule Base

The design of the rules is application-dependent but it should aim to achieve the following attributes.

Complete: A set of rules is complete if any combination of input value results in an appropriate value. Any combination should fire at least one rule.

Consistent: There should be no contradiction in the rules.

Continuous: There are no neighbouring rules with output fuzzy sets that give empty intersections or in other words, small changes in the inputs should result in smooth change in output.

Since a particular input may fall within more than one component of a fuzzy set, multiple rules may be true for any given set of inputs. Each cell in the FAM contains the linguistic terms for the corresponding input combination such as PL (positive large), PM (positive medium) and NL (negative large). In this step, each rule fires to its particular degree, and we obtain an output fuzzy set, which represents the result of the inference. This is, usually, scaled by the degree to which the corresponding rule fired, and derived by adding the resulting functions up. This gives us a new membership function, which defines a fuzzy set corresponding to an output of the system. This is done for every output.

Defuzzification

The final step is called defuzzification. Since in most cases we are interested in a crisp output, such as the motor speed for an air conditioner, we need to convert the resulting output fuzzy sets into one crisp value. There are various ways of doing this, such as the max criterion method, mean of maximum method and centroid method [Lee 1990]. The output fuzzy set relates each output linguistic variable to a numerical value (Singleton

defuzzification) [Kosko1994] For example, the linguistic variables NL and NM could be assigned numerical values of -5 and -3, respectively.

2.5. Summary

In this chapter we formulated the personalisation of information retrieval as a problem of user modelling in order for it to be adaptive. We surveyed the literature of machine learning techniques specially: fuzzy logic, Genetic Algorithms, relevance feedback and proposed that combining their strengths would be a novel approach to demonstrate adaptive behaviour. The use of genetic algorithms and fuzzy logic is proposed in the Chapter, and the fundamental concepts of both techniques are reviewed.

Chapter 3. INFORMATION RETRIEVAL

3.1. Introduction

Information comes in many forms: news, financial data, scientific research, etc. It represents one of the most important commodities in the modern world today. All major political, economic, and scientific activities rely on the quick and easy flow of information through a variety of media, such as books, radio, television and electronic network [Gudivada et al., 1997]. Modern computing and networking technologies make it possible to organise, store and transport large bodies of data with minimal effort anywhere in the world. With so much material, many researchers have realised that the real issues is no longer getting information, but sorting out what is useful to them from vast quantities of irrelevant material. Information retrieval (IR) deals with such problems. IR systems have been in existence for many years and are coming more and more into focus, not least with the developments in the WWW. Although the data can consist of any media type, this research focuses only on the retrieval of text documents.

The discipline of information retrieval is almost as old as computer technology itself. The following definition of information retrieval was given by Mooers [1950]:

> *"Information retrieval is the name of the process or method whereby a prospective user of information is able to convert his need for information into an actual list of citations to documents in storage containing information useful to him".*

As information retrieval some times mixed with information filtering, the following section represents the difference between them.

3.2. Information Retrieval and Information Filtering: A Comparison

Information Filtering is a name used to describe a variety of processes involving the delivery of information [Belkin and Croft 1992]. A more specific definition describing the Information Filtering (IF) problem is given by Wondergem et al. [1997]:

> *"Considering a number of dynamic information objects, the IF system matches the characterizations of the information objects against the user profiles, descriptions of the users' information needs, to obtain a relevance estimate of the information objects with respect to the information needs."*

Belkin and Croft [1992] raise the question, whether IR and IF are *"two sides of the same coin"*, insinuating a connection between the two disciplines. Both disciplines deal with the same objective, satisfying the information needs of people, using similar methods. The main differences between them are the nature of information need and the information source. Information retrieval has normally been associated with systems receiving dynamic and specific queries to search for information from stable and unstructured information sources. Information filtering, in contrast, refers to systems that match a user's long-term goal against dynamic information sources. An information retrieval system normally satisfies the user's interest within a single session, while information filtering systems normally involve long-term interests and multiple interactions over repeated sessions. Ideally, the filtering system should eliminate irrelevant results while keeping relevant results for the user [Baeza-Yates 1999]. The primary differences are summarised in Table 3-1.

Four kinds of filtering can be distinguished:

- **Cognitive filtering** -selection according to the content, often determined by index terms, including natural language.
- **Economic filtering** -selection according to formal criteria related to resources and costs, for example, the length of a document.
- **Social filtering** - Selection of documents by determining what documents other users with similar interests and/or needs found relevant. This often works as a recommendation system, for example, recommendations for music, movies and news.

A simple filter determines, using some external criteria, whether to let the information through, or where to put it (examples are the filters for email applications). A filtering system has a user profile, which is a model of what information its user might need (user's preferences). Such a profile is then compared to the incoming documents in an attempt to determine those which might be of interest.

	Information Retrieval	Information Filtering
User	Ad-hoc interests, mostly by an individual	Regular information interests or needs on the same topic, often by groups
Query	Ad-hoc query according to actual information need	Determined by the information and related profile of group
Seeking process	Active	Passive
Time	Short-term	Long-term
Privacy	Not much attention paid to	Problem as user profile may need protection
Source of information	Mostly fixed (e.g. Database)	Data streams
Level of user interaction	High (e.g. formulating query, relevance feedback)	Medium (definition of user profiles and query terms)
Volume of data	Variable	Mostly huge
Paradigm	"Finding" information in a stream	"Removing" information from a stream

Table 3-1: Comparison of IR and IF

3.3. The Information Retrieval Process

In information retrieval, the challenge is to retrieve relevant documents in response to user queries [Mooers 1976]. An information retrieval system is a software program that stores and manages information on documents. The objective of such systems is to assist users in finding the information they need. Some suggested documents will, it's hoped, satisfy the user's information need. These documents are called *relevant* documents [Witten et al., 2000]. A perfect retrieval system would retrieve only the relevant documents and no irrelevant document. However, perfect retrieval systems do not exist and will not exist because search statements are necessarily incomplete and their relevance depends on the subjective opinion of the user.

The two basic approaches to defining relevance are:

1. *Objective*: where a query and a document are given and the contextual concordance of the query and the document are the different levels of relevance [Salton and McGill 1997]. Here relevance is independent of the knowledge of the information seeker, but documents previously seen are also relevant.

2. *Subjective*: here a document is relevant if its content is new to the information seeker [Salton and McGill 1997]. This means that the information seeker as an individual and his knowledge about an information need to be looked at. Having these two definitions in mind, it is clear that from the point of view of a user, subjective relevance is needed. Two users may pose the same query to an information retrieval system and give different relevance judgments on the retrieved documents.

Let's start with an abstract view of the working of an IR system. When thinking of the IR process (model), different points of view can be taken, but all models have a common goal, which is finding those documents that are relevant to a user's query. Van Rijsbergen [1979] illustrates a typical IR model (Figure 3-1).

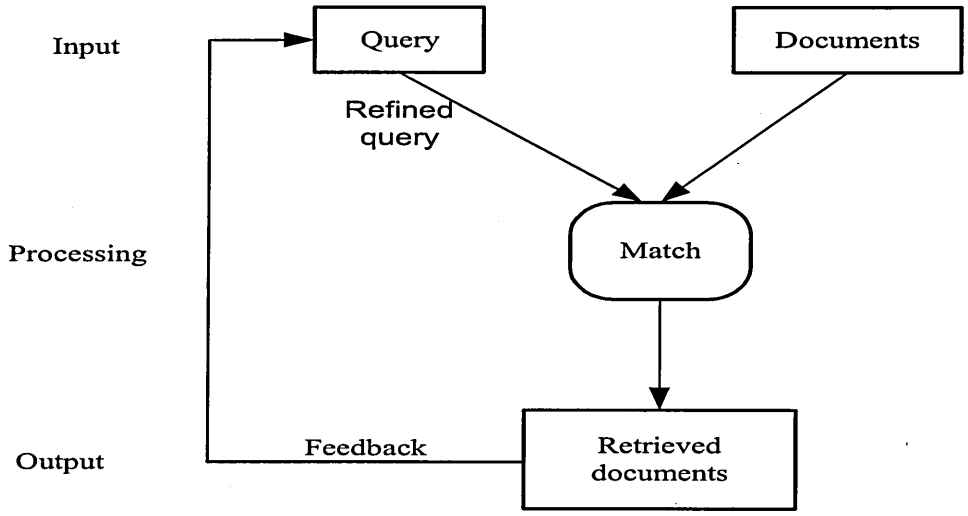


Figure 3-1: Typical model of an information retrieval system

Croft presents a more sophisticated model than the black-box by describing the basic entities and processes of IR situations [Croft 1993] as shown in Figure 3-2. Square boxes represent data and rounded boxes represent processes.

Here, retrieval starts with a user query (an information problem). This information problem is converted into an internal representation to allow it to be processed by the system. This transformed query is then matched against a collection of documents also stored in the same representation. The system assigns to each document in this collection a score that indicates the relevance of that document to the supplied query. The documents in this collection are ranked by their assigned *relevance score*, with the highest scoring documents being presented in rank order to the user. Based on the above model there are three basic processes that an information retrieval system has to support: the representation of the content of the documents, the representation of the user's information need, and the comparison of the two representations [Croft 1993].

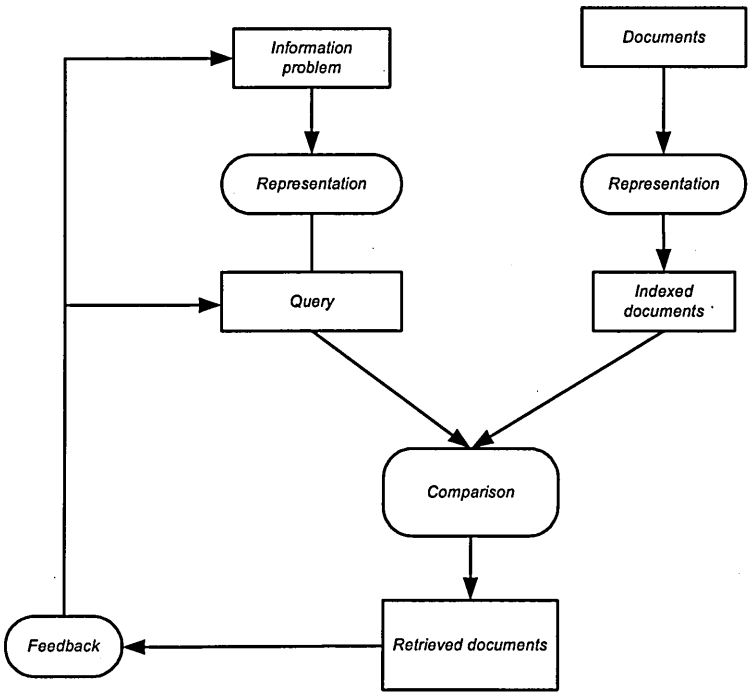


Figure 3-2: Information Retrieval Process

3.3.1. Document Representation

Representing a document is usually called an '*indexing process*'. The representation typically used by IR systems is a set of features derived from the document collection. Each document in the collection is represented as a list of its features. The query is also represented in a similar manner. The feature set most commonly used, is the set of words in the document collection. A number of pre-processing steps are involved before words become feature terms, such as,

- Words will typically have the case of their letters normalised.
- Certain types of words such as prepositions, determiners, pronouns, will be removed from the feature set. Such words are known as *stop words.* The use of such lists decreases the size of the indexing terms considerably, but their elimination might lead to a lower recall [Baeza and Ribeiro-Neto 1999].
- Words are stemmed to replace them with their root forms. There are two methods for stemming: rule-based [Nahm and Mooney 2001] and automatic suffix algorithms [Porter 1997].

Once the words have been processed into a feature set they are referred to as terms [Frakes 1992, Witten et al., 2000]. An additional subtlety to the representation of documents is the assignment of a numerical weight to all terms in a document collection. The weight assigned to a term occurring in a document is an attempt to quantify that term's importance to the subject of that document.

3.3.2. Query Representation

The process of representing an information-needs problem is often referred to as a *query formulation* process [Baeza and Ribeiro-Neto 1999]. The resulting formal representation is the query. In a broad sense, query formulation might denote the complete interactive dialogue between the system and user, leading not only to a suitable query but possibly also to a better understanding by the user of his/her information need. Relevance feedback uses a query or request, and some previously retrieved relevant and non-relevant documents to generate a successive query. A firing Query model in which information flow is computed based on the Hyperspace Analogue to Language (HAL) is an approach to query formulation proposed by Bruza and Song [2002].

3.3.3. Matching Process

The comparison of the query against the document representations is called the *matching* process. The matching process results in a list of potentially relevant documents. Users will browse this document list in search of the information they need. The objective of ranked retrieval is to put the most relevant documents in the top of the ranked list, minimizing the time the user has to invest in reading the documents. Simple, but effective ranking algorithms use the frequency distribution of terms over documents [Salton and McGill 1983]. With respect to this Baeza and Ribeiro-Neto [1999] see ranking algorithms at the core of IR systems. There are many kind of models can be used for the matching process including the binary model, the vector space model that and the probabilistic model. These models are so called approximate matching models, because they use the frequency distribution of terms over documents to compute the ranking of the retrieved sets. Each of these models has its own advantages and disadvantages which will be discussed in the next section. There is another method of ranking hypertext pages in web pages e.g. Google ranking method. Google has a very effective ranking algorithm called PageRank which attempts to give more important or higher quality web pages a higher ranking. The basic idea behind the PageRank algorithm is to use the number of links to a web page as a source for ranking. Highly linked pages are ranked more important than pages that do not have as many links to them. The links themselves are divided into a set of backlinks and a set of forward links. Backlinks are links that link or point to a certain web page. Forwards links are links that a web page points to. For example, as of November 2003, the Google web page has over 300,000 web pages linking to it. Hence, it can be assumed that the Google web site is an important and credible web page. Though this is the basic idea behind the algorithm, the number of backlinks alone cannot guarantee a good ranking. The ranking also depends on the rank of the page that is linking to it. For example, if a web page has only one backlink, but that backlink is from a credible web page such as the Stanford University home page, the web page would then be given a high ranking [www.google.com].

3.4. Information Retrieval Models

Information retrieval models are a prediction of what is relevant and what is not, and also guide the implementation of information retrieval systems. Figure 3-3 shows a classification of IR models [Baeza and Ribeiro-Neto 1999].

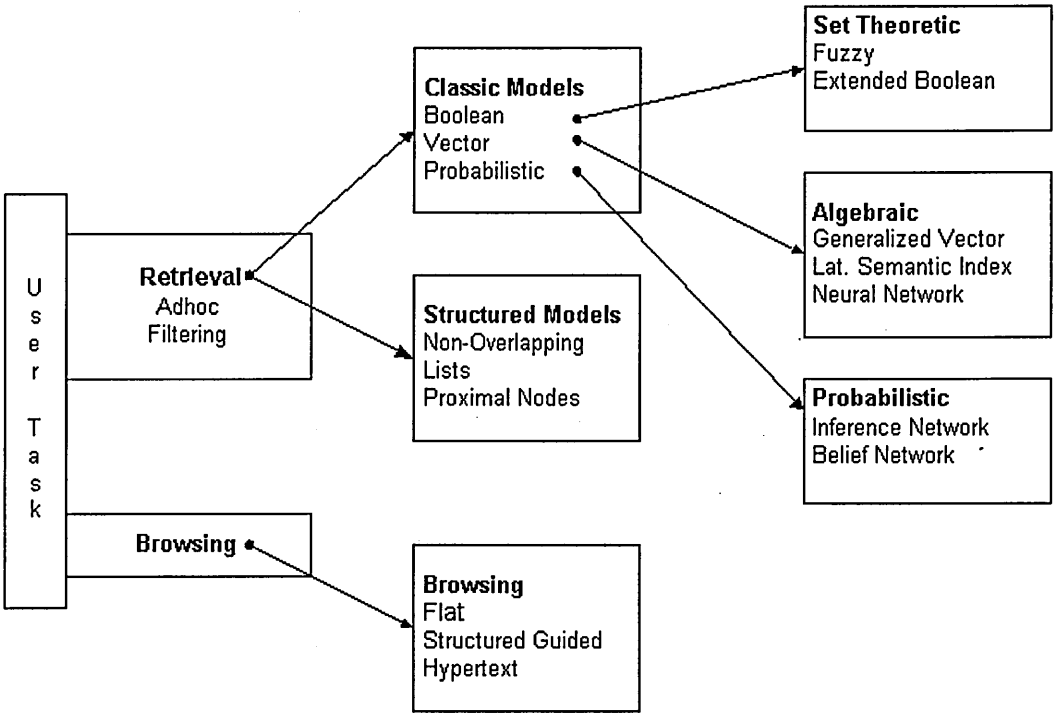


Figure 3-3: Taxonomy of IR models

In the Boolean model, documents and queries are represented as sets of index terms. Thus, it is also called a set theoretic model. The vector space model is characterised by representing documents and queries as vectors in a t-dimensional space. Thus, the model is called algebraic. In the probabilistic model, document and query representation is based on probability theory [Baeza and Ribeiro-Neto 1999, Charles 1999]. Some extensions for each type of classic model have been proposed (Figure 3-3) such as:

- Fuzzy and extended Boolean, which are set-theoretic models.
- Generalised vector, latent semantic indexing and neural networks, which are algebraic models.
- Inference network and belief network, which are probabilistic models.

These basic models are discussed in more detail in following subsections.

3.4.1. Boolean Model (Exact match model)

The Boolean model has been extensively used as the basic model of information retrieval, but is probably also the most criticized model. This model is a simple retrieval model based on set theory and Boolean algebra. The model can be explained by thinking of a query term as a definition of a set of documents. For instance, the query term k_a simply defines the set of all documents that are indexed with the term k_a. Using Boolean operators, query terms and their corresponding sets of documents can be combined to form new sets of documents. Boolean logic defines three basic operators, the logical product, AND, the logical sum, OR and the logical difference, NOT. Combining terms with the AND operator will define a document set that is smaller than or equal to the document sets of any of the single terms. For instance, the query k_a AND k_b will produce the set of documents that are indexed with both the term k_a and the term k_b. Combining terms with the OR operator will define a document set that is bigger than or equal to the document sets of any of the single terms. So, the query k_b OR k_c will produce the set of documents that are indexed with either the term k_b or the term k_c, or both. This is illustrated in the Venn diagrams shows in Figure 3-4. The intersections of these sets and their complements divide the document collection into eight regions, the unions of which give 256 different Boolean combinations of 'k_a, k_b and k_c documents'. The shaded areas in the figure illustrate some of the possible retrieved sets.

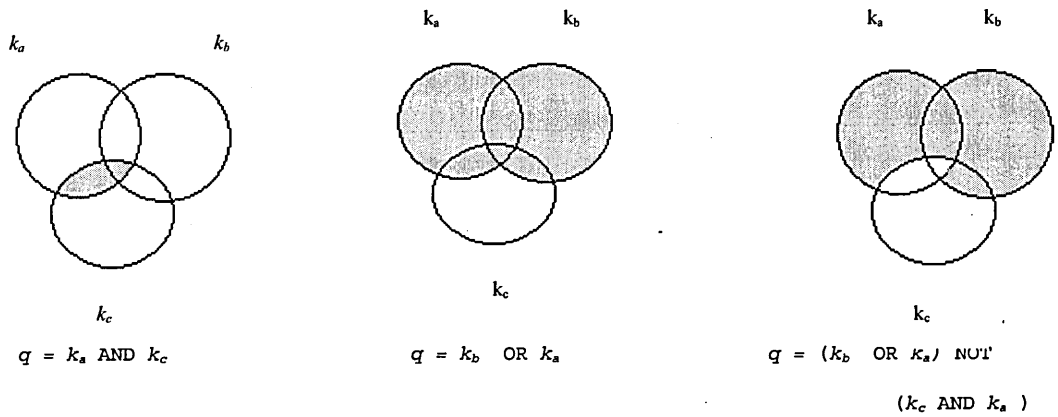


Figure 3-4: Three Boolean combinations of sets visualized as Venn diagrams

The main advantages of the Boolean model are the clear formalism behind the model and its simplicity. The main disadvantage is that exact matching may lead to retrieval of too few or too many documents [Baeza-Yates 1999, Charles 1999].

3.4.2. Vector Space Model

The vector space model (VSM) recognizes that the use of binary weights is too limiting and proposes a framework in which partial matching is possible. This is accomplished by assigning non-binary weights to index terms in queries and in documents [Charles 1999]. Salton and McGill [1997] suggested a model, which represents queries and documents as vectors of terms, with associated weights representing the importance of each term inside the documents.

The idea behind term weighting is selectivity: what makes a good term is whether it can pick any of the few relevant documents from the many non-relevant ones. The three sources of weighting are [Salton and Buckley 1998]:

- *Collection Frequency* – Terms which occur in only a few documents are likely to be more useful than ones occurring in many.
- *Term Frequency* - The more frequently a term appears in a document, the more likely it is to be relevant for that document.
- *Document Length* - A term that occurs the same number of times in a short document as in a long one is likely to be more important to the short document than it is to the long one.

The weight assigned to a term occurring in a given document is an attempt to quantify that term's importance to the subject of that document. There are many methods for calculating the weight of a term [Frakes 1992]. Most are statistical, based on the term's frequency of occurrence within a collection of documents, known as the inverse document frequency (idf), and on its frequency of occurrence within the document, known as the term frequency (tf). The term weighting function, shown in equation 3-1, is typical of such methods. The weight resulting from this function is often referred to as a tf•idf weight [Salton and McGill 1997]:

Let D = $\{d^1, d^2, \ldots\ldots, d^n\}$ denote a set of documents and T=$\{t_1, t_2, \ldots\ldots, t_m\}$ a set of all terms of the documents. The weight $w^j{}_i$ of a term t_i in document d^j is given by [Salton and McGill 1997]:

$$w_i^j = \frac{1}{2}(1 + \frac{f_i^j}{f_{\max}})(\log\frac{n}{n_i}) \qquad (3\text{-}1)$$

where $f^j{}_i$ is the number of times term appears in document d^j (the *term frequency*), n_i is the number of documents in the corpus which contain t_i (*the document frequency*), n is the total number of documents in the corpus and f_{max} is the maximum term frequency over all terms in the document. The above weight function can be normalized to:

$$w_i^j = \frac{f_i \log(n/n_i)}{\sqrt{\sum_{j=1}^{n}(f_i)^2[\log(n/n_i)]^2}} \qquad (3\text{-}2)$$

If terms are assumed to be independent one can, for example, estimate the similarity between a query and a document by the inner product of their corresponding vectors. Various functions to measure this similarity exist. A commonly used similarity function is the cosine (equation 3-3), which measures the cosine of the angle between the document vectors and query vector, where values closer to 1.0 indicate a higher similarity [Salton and Buckley 1997a]. The result is a similarity measure for each document that indicates how closely it matches a query. Documents are finally ranked by the similarity values and the ones with higher values are then selected as the result of the query. The similarity measure can be formalized as an inner product of a query vector with a set of document vectors, given by:

$$Sim(\mathbf{a},\mathbf{b}) = \frac{\sum_{j=1}^{n}\sum_{k=1}^{n} w_{aj} \cdot w_{bk}}{n\sqrt{\sum_{j=1}^{n}\sum_{k=1}^{n}(w_{aj})^2(w_{bk})^2}} \quad \forall\,(t_j = t_k) \qquad (3\text{-}3)$$

Where w_{aj} are the weights of terms in the query vector (**a**) and w_{bk} are weights of similar terms in the document vector (**b**).

The main disadvantage of the vector space model is that, it is not possible to include term dependencies into the model, for instance, for representation of phrases or adjacent terms. This problem has led to a new form of the vector space model, called the Set-Based model, in which the components are no longer terms, but term sets [Possas et al., 2002]. On other hand, the vector space model has its advantages, such as [Charles 1999]:

- Term weighting improves retrieval performance.
- Partial matching allows retrieval of documents that approximate the query conditions and document.
- Ranking sorts the documents according to their degree of similarity to the query.

3.4.3. Probabilistic Models

This model was first developed by Steve Roberston and Karen Sparck in the 1970s, and later became known as *binary independence retrieval* [Robertson 1997]. Roberston adopted the Boolean model's viewpoint by looking at a term as a definition of a set of document, with the index term weight all binary. The probabilistic model is one of the few retrieval models that do not need an additional term weighting algorithm to be implemented. It relies on ranking algorithms that are completely derived from probabilistic theory [Losee 1998]. The probability model has been one of the most influential retrieval models for this very reason. Unfortunately, in many applications, the distribution of terms over relevant and non-relevant documents will not be available. In these situations, probability of relevance estimation is of theoretical interest only.

The probabilistic approach ranks documents in the collection in order of their *probability of relevance*, so called the "probability ranking principle". The model is based on the following principle [Robertson 1997]:

Given probability for a particular query that a document is relevant $P(R \mid \dot{d})$ and given a user query q (a subset of index terms) and document $\dot{d}$ in the collection, the model tries to estimate the probability that the user will find the document relevant. It assigns the similarity value, $sim(\dot{d}, q)$ of the document $\dot{d}$ to the query q using the formula:

$$sim\ (d^{j}, q) = \frac{P(R|d^{j})}{P(R'|d^{j})} \qquad (3\text{-}4)$$

Where R is the set of documents known (or initially guessed) to be relevant.

R' is the set of non-relevant documents.

$P(R \mid \vec{d})$ is the probability that the document $\vec{d}$ will be relevant to query q.

$P(R' \mid \vec{d})$ is the probability that the document $\vec{d}$ will be non-relevant to query q.

By using Bayes' rule the above formula can be written as:

$$sim\ (d^{j}, q) = \frac{P(R|d^{j}) \times P(R)}{P(R'|d^{j}) \times P(R')} \qquad (3\text{-}5)$$

One of the main problems with this model is, the method to estimate $P(R|\vec{d})$ and $P(R' \mid d)$. Probability estimation based on relevance feedback is a common solution [Robertson 1997].

The main disadvantage of the probabilistic model is that it only defines a partial ranking of the documents [Charles 1999]. For short queries, the number of different subsets will be relatively low. By looking at a term as a definition of a set of documents, the probabilistic model ignores the distribution of terms within documents. In fact, one might argue that the probabilistic model suffers to some extent from the same deficiency as the Boolean model; it does not allow the user to really control the set of retrieved documents. For short queries it will, most of the time assign the same rank to, the first retrieved documents. However, in a recent extension of this model, term weighting was considered [Greiff et al., 2002]. In particular, the importance of variance in term weighting was proposed. Other probabilistic approaches have been suggested including, probabilistic neural networks [Kwok 1995].

3.4.4. Fuzzy Models

In fuzzy set theory [Zadeh 1965], an element has a continuous degree of membership to a set (section 2.6). Whereas in the Boolean IR model documents either belong to the set defined by an index term or not, in the fuzzy set model documents belong with a certain

degree to the set defined by an index term. The degree of membership is used to represent inexactness or vagueness. Although it can be known with certainty that a number of documents contain a specified term, some of the documents will be more relevant than others [Charles 1999]. For the degree of membership T of a single term, the document term-weighting formula (3-6) of the vector space model can be used. The rules for the membership function T of the union and intersection of fuzzy sets are then as follows:

$$\begin{aligned} &T(\text{a AND b}) = \min(T(\text{a}), T(\text{b})) \text{ or } T(\text{a})*T(\text{b}) \\ &T(\text{a OR b}) = \max(T(\text{a}), T(\text{b})) \text{ or } T(\text{a})+T(\text{b})-T(\text{a})*T(\text{b}) \\ &T(\text{NOT b}) = 1 - T(\text{b}) \end{aligned} \qquad (3\text{-}6)$$

Although the fuzzy set model, which is a multi-valued logic extension of the Boolean model, overcomes the problem of the conventional Boolean retrieval system, it has been shown to generate incorrectly ranked output in certain cases. This is because the *min* and *max* operators used for "AND" and "OR" operations do not correspond well with human approaches for document ranking. These operators are not very effective for single operand dependency and negative compensation [Lee 1994]. Suppose for a query "a OR b ", we obtain T(a) =0.8 and T(b) = 0.7 in one document and, T(a) = 0.8 and $T(b)$ = 0.1 in a second document. Both documents would be ranked equal, using the *max* operation. Intuitively, one would rank the first document above the second document in the example. A similar argument can be constructed for the intersection of fuzzy sets.

The Wailer-Kraft, Paice, P-Norm and Infinite-One models have been developed to overcome the problem of the fuzzy models [Paice 1984]. Paice's set operator takes into account all the document weights in the final score, not only the maximum or minimum weights. The score of a document given a query (a_1 AND a_2 AND $\cdots$ AND a_n) or a query (a_1 OR a_2 OR $\cdots$ OR a_n) is computed as follows:

$$score = \frac{\sum_{k=1}^{n} r^{i-1} T(a_k)}{\sum_{k=1}^{n} r^{i-1}} \qquad (3\text{-}7)$$

where the $T(a_k)$ are considered in descending order for OR queries and in ascending order for AND queries. For Boolean queries with more than one operator, the evaluation proceeds recursively from the innermost clause. The value of r has to be determined experimentally for both set operators. It determines the 'softness' of the operator. For values close to 1.0 the operators show similar behaviour.

One of the applications of fuzzy models in IR is the adaptation of thesaurus, which is to expand the set of index term in a query with related terms [Baeza and Ribeiro-Neto 1999]. Another application is in fuzzy indexing models of WWW structured documents, which allow users to tune the representation of documents [Bordogna and Pasi 2002]. Fuzzy models have also been used for vocabulary mining to improving the user's query [Srinivasan et al., 2001]. Fuzzy set models have the advantage over the vector space model and the probabilistic model in that they provide a ranking for structured queries. An extensive comparison, both in terms of theoretical properties and retrieval effectiveness of fuzzy set models and other extended Boolean models was conducted by [Lee 1994].

Chau and Yeh [2000] proposed an explorative approach to multilingual text retrieval (MLTR) based on fuzzy multilingual keyword classification. The approach applies fuzzy clustering to obtain a classification of multilingual keywords by concepts. A multilingual concept directory is developed by labelling each concept with native language of the target user and associating it with relevant multilingual documents. The other application is using a fuzzy indexing model of WEB structured document that allows users to tune the representation of documents [Bordogna and Pasi 2002]. Fuzzy models have also been used for vocabulary mining to improving the user's query [Srinivasan et al., 2001].

3.5. Information Retrieval Using Genetic Algorithms

In IRS, the purpose of using GA is to help to find, in a huge text collection, a good reply to a query expressed by the user, employing adaptive behaviour. Information retrieval using genetic algorithms is based on vector space model. Within this model, both

documents and queries are represented by vector. A particular document is represented by vector of terms and a particular query is represented by vector of query terms.

A document vector (Doc) with n keywords and a query vector with m query terms can be represented as

$Doc = (term_1, term_2, term_3 \ldots . term_n)$

$Query = (qterm_1, qterm_2, qterm_3, \ldots . . qterm_m)$

We use binary term vector, so each $term_i$ (or $qterm_j$) is either 0 or 1. $Term_i$ is set to zero when $term_i$ is not presented in document and set to one when $term_i$ is presented in document. For example, user enters a query into the system that could retrieve 5 documents. These documents are

Doc_1 = {Relational Databases, Query, Data Retrieval, Computer Networks, DBMS}

Doc_2 = {Artificial Intelligence, Internet, Indexing, Natural Language Processing}

Doc_3 = {Databases, Expert System, Information Retrieval System, Multimedia}

Doc_4 = {Fuzzy Logic, Neural Network, Computer Networks}

Doc_5 = {Object-Oriented, DBMS, Query, Indexing}

All keywords of these documents can be arranged in the ascending order as **Artificial Intelligence**, Computer Networks, **Data Retrieval**, Databases, **DBMS,** Expert System, **Fuzzy Logic**, Indexing, **Information Retrieval System**, Internet, **Multimedia**, Natural Language Processing, **Neural Network**, Object-Oriented, **Query**, Relational Databases

Encode in the chromosome representation as

Doc_1 = **0 1 1** 0 1 **0 0 0** 0 0 0 0 **0 1** 1

Doc_2 = **1 0 0** 0 0 0 **0 1** 0 1 0 1 **0 0 0** 0

Doc_3 = **0 0 0** 1 0 1 **0 0** 1 0 1 0 0 **0 0 0** 0

Doc_4 = **0 1 0** 0 0 0 **1 0** 0 0 0 0 **1 0 0** 0

Doc_5 = **0 0 0** 0 1 0 **0 1** 0 0 0 0 0 **1 1** 0

These chromosomes are called initial population that feed into genetic operator process. The length of chromosome depends on number of keywords of documents retrieved from user query.

FITNESS EVALUATION

Fitness function is a performance measure or reward function which evaluate how good each solution is. The information retrieval problem is how to retrieve user required documents. The fitness functions in Table 3-2 to calculate the distance between document and query. From Table 3-2, there are 2 types of fitness functions: weighted term vector and binary term vector.

$X = (x_1 , x_2 , x_3 ,\ldots.., x_n)$, $|X|$ = number of terms occur in X , $|X \cap Y|$= number of terms occur in both X and Y [Salton 1989].

Similarity Measure Sim (X,Y)	**Binary Term Vectors**	**Weighted Term Vectors**
Dice coefficient	$2\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert}$	$\frac{2\sum_{i=1}^{t} x_i . y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2}$
Cosine coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert^{1/2} . \lvert Y \rvert^{1/2}}$	$\frac{\sum_{i=1}^{t} x_i . y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 . \sum_{i=1}^{t}}}$
Jaccard coefficient	$\frac{\lvert X \cap Y \rvert}{\lvert X \rvert + \lvert Y \rvert - \lvert X \cap Y \rvert}$	$\frac{\sum_{i=1}^{t} x_i . y_i}{\sum_{i=1}^{t} x_i^2 + \sum_{i=1}^{t} y_i^2 - \sum_{i=1}^{t} x_i . y_i}$

Table 3-2: Fitness Function

Result from these fitness functions are between 0 to 1. Values near 1.0 mean documents and query are more relevant and values near 0.0 mean documents and query are less relevant. Values evaluate from fitness functions are called "fitness".

Literature reviews revealed several recent implementations of genetic algorithms in information retrieval. In [Gordon 1988], presented a genetic algorithms based approach for document indexing. Competing document descriptions (keywords) were associated with a document and altered over time by using genetic mutation and crossover operators. In his design, a keyword represented a gene (a bit pattern), a document's list of keywords represented individuals (a bit string), and a collection of documents initially judged relevant by a user represented the initial population. Based on a Jaccard's score matching function (fitness measure), the initial population evolved through generations and eventually converged to an optimal (improved) population - a set of keywords which best described the documents.

[Raghavan and Agarwal 1987] also studied genetic algorithms in connection with document clustering. In [Petry et al. 1993], applied genetic programming to a weighted information retrieval system. In their research, a weighted Boolean query was modified in order to improve recall and precision. They found that the form of the fitness function had a significant effect upon performance.

Finally GA has been used mainly for two different applications in IR: document indexing [Vrajitoru 1998] and improvement of query formulation [Chen and Shankaranarayanan 1998, Smith 2002]. Horng and Yeh [2000] proposed an approach, which uses genetic algorithms used to adapt term weights.

The major problem encountered in the use of GA to solve any problem is the appropriate choice of underlying operators, such as encoding, definition of fitness function, operators (reproduction and mutation) and their parameters (mutation probability, crossover probability and population size) [Goldberg 1989]. In this research, GA are used in user-modeling to evolve an optimal set of vectors (documents) that best match the user's needs (expressed in terms of keywords and their weights). This is done by generating queries that can sufficiently identify relevant documents and reject irrelevant documents. User models, in this case, represent knowledge about the user's interests, encoded in a chromosome. Each chromosome is a hypothesis on how to evaluate a document according to the information provided, and competes to predict user satisfaction from the document.

One common way to represent genes in IR is binary encoding where each bit in the chromosome represents a certain keyword. The location of a gene decides the existence

(1, ON) or non existence (0, OFF) of the keyword [Vrajitoru 1998, Aizawa 2002]. In the present study, however, each gene of a chromosome is specified by a pair (t, w), where t is a term (a word extracted from a document) and w is the term's weight, calculated based on inverse document frequency method (section 3. 4. 2) [Salton and Buckley 1998]. The goal of a GA in the system is to find the optimal set of documents that best matches the users' interest.

3.6. Summary

In this chapter we proposed that adaptive user modelling could be achieved via an adaptive system for information retrieval and therefore surveyed the literature in IR to capitalise on an overview of information retrieval models, namely, Boolean, vector space, probabilistic and fuzzy models were presented. Pertinent issues including document representation, query representation, and the query-document matching process were discussed. The feasibility and applicability of user-needs modelling was also discussed. Contemporary paradigms for information retrieval and filtering were compared. Finally the Chapter discussed genetic algorithms and fuzzy logic application in information retrieval systems.

Chapter 4. AGENT TECHNOLOGY

4.1. Introduction

The concept of agents primarily comes from Artificial Intelligence (AI) research. Its aim is to develop a technique that provides intelligent capabilities, similar to human reasoning and learning. These such techniques has been established years ago and been used in various intelligent application. However, such system is not able to show the real intelligent as human do such as while he was alive and having up datable knowledge and make decision. The current expert system only helps on helping keeping massive knowledge while make decision but still depend to the user. This system is not able deal with dynamic environment. The agent technology is able to provide a real intelligent compare to expert system. The three criteria of agent is autonomy, distributed, flexible and intelligence. With such techniques, the computer's role tends to depart from that of a mere tool and progressively becomes an assistant and advisor to humans in problem-solving applications, especially where distributed real time applications are employed [Jenning and Wooldrige 1998, Sarma 1996]. Therefore, work in the agents field has moved to focus not only on intelligence, as defined in the AI community, but has been enhanced by blending other technologies, such as, network communication [Harries 1993], distributed problem-solving [Lesser and Crokill 1987], distributed systems [Mullender 1993], concurrency [Ranky 1994], knowledge engineering [Adeli 1990], distributed artificial intelligence [Bond and Gasser 1988] and object-oriented technology [Booch 1994]. With the combination of such technologies, agent-based solutions represent a useful and attractive technology as a solution to several kinds of complex applications which having following criteria [Wooldridge, 2001]:

- The environment is open or at least highly dynamic, uncertain or complex. In such environments, the system having flexible autonomous action is the only solution.

- Agent are a natural metaphor. Many environments (including most organisations and any commercial competitive environment) are naturally modelled as societies of agents.

- Distribution of data, control or expertise. In some environment, the distribution of either data, control or expertise which is not appropriate centralise solution. For example, distributed unstructured and different type of database which link together to provide meaningful information.

- Legacy systems, the nature of current organisation is having various independent of legacy systems which cannot generally be discarded. And yet it is often required to interact with other software components. Agent technology solution is to *wrap* the legacy components and providing them with agent layer functionality, enabling them to communicate and cooperate with other components.

Agents allow software developers and system designers to use high-level abstractions in building software to manage complexity. An abstraction focuses on the important and essential properties of a problem and hides the incidental components of that problem. Agents provide a new way of managing complexity because they provide a new way of describing a complex system or process. Using agents, it is easy to define a system in terms of agent-mediated processes. Agents provide a special communication technology that allows an agent to communicate with other agents using a speech-act language. The use of this language mimics human communication. In addition, it allows the system developer to reuse existing communication protocols and message formats and the use of such languages provides openness, in that agents are not limited to a specific language. This capability is provided as part of the basic agent mechanism. Agents have the inherent capability to build models of their environment, monitor the state of that environment, reason and finally make decisions based on that state.

Agents are well suited for applications with a high degree of independence, such as, distributed decision making system, but, have dependency attributes that allows them to share knowledge through communication. Therefore, agent technology is well suited for applications that involve messages or objects transmitted and received over a network. Agent technology provides powerful tools and techniques, which are being used in an

increasingly wide variety of complex applications, such as, manufacturing, process control, information filtering and electronic commerce [Brenner et al., 1998; Wooldrige and Jenning 1998]. Outside the AI community, the agent concept has been used in other areas as well, such as, object-oriented programming (OOP) and human computer interaction (HCI) [Jenning et al., 1998]. The technology provides a new way of thinking about design solutions for development of software applications. It provides a capability to delegate the functions of a complex system in such a way as to reduce human effort. This is discussed in more detail in Chapter 5.

There are other properties, which can be discussed in the context of agency (ability to make decisions through reasoning) , such as, learning (ability to change its behaviour based on past experience), intelligence, mobility (ability to move around in a computer communication network), veracity (not knowingly communicating false information), benevolence (the assumption that agents do not have conflicting goals), rationality (acting to achieve its goal and not preventing their achievement, selectivity (ability to focus attention on what it needs and ignoring the rest) and robustness (ability to cope with failures and tolerate imperfection) [Jenning et al., 1998]. Based on the above descriptions, Brenner classified various areas that exploit these agent characteristics and definition as shown in Figure 4-1[Brenner et al., 1998].

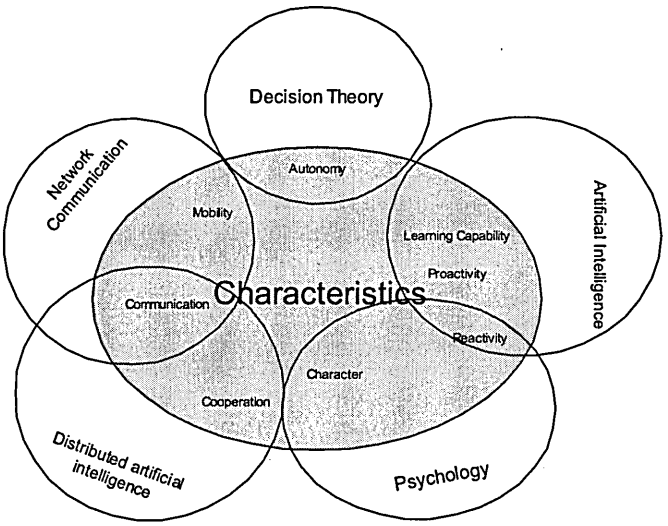


Figure 4-1: Agents areas of influence

Exploiting those agent definitions [Bigus and Bigus, 1998] have outlined the three types of processing strategies that agents may posses either individually or as a group:

- *Reactive agents*- simple reflex input-process-output systems, low level data are passed, without high level knowledge.
- *Pro-active agent*- autonomy is taken to be at its extreme; it requires some knowledge of domain and ability to plan the actions that needs execution.
- *Collaborative agent* - communication between the agents is crucial, as individually the agents may not be skilled or knowledgeable enough.

Reactive agent responds to the changes made by the agent's environment or actions take due to the change of internal state of agent or influence by other agent. This requires the agent to react only to stimuli and as such a symbolic representation of the world is not required. These types of agents will tend to be simple reflex input-process-output systems as demonstrated in the use of robots perceiving the world in terms of their sensors. Where there is interaction between agents, only low level data is passed between them; not high level knowledge, which is limiting for complex MAS. Finlay and Dix [1996] demonstrate a simple use of reactive agent technology. They described an e-mail sorting agent as an example of a reactive agent. It starts, upon receiving a message, by looking for specific key-words, such as, seminar names or addresses, and sorts the messages into boxes of importance to the user.

Pro-activeness has already been suggested to be a characteristic of an agent, but taking this a step further; a proactive agent may be able to demonstrate self co-ordination to achieve its goals, at the extreme, without interaction with humans. This "higher intelligence" requires the agent to have some knowledge of the domain and also to be able to plan the series of actions needed to execute to achieve the goal.

Collaborative agents work together to solve a problem. Communication between the agents is crucial, enabling achievement of the goal through the synergy of cross-agent co-operation. Individually, the agents may not be skilled or knowledgeable enough to solve the whole problem, but together they can share the work of the agents within the group whose aim it is to achieve the same goal. Despite several processing strategies an agent should possess at least the first two of the above properties to present a degree of

usefulness of the agent paradigm in development of software applications [Jenning and Wooldridge 1998].

4.2. Agent Architecture

Agent architecture is the structure that supports the characteristics or properties of agents when constructed on a computer system. The structure is illustrated by presenting the components and their inter-relationships. In other words, agent architecture can be thought of as a software engineering model for agents. Several types of agent architecture have been developed for agent system solutions. This architecture can be reused for similar problem domains. Each type of agent as presented in the previous section has its own architecture. Some solutions present a combination of agent types, for example, the use of mobile agent and Information agent for a solution of filtering articles from the Web. The following is a description of three common architectures: deliberative, reactive, and hybrid architectures [Berrener and Zarnekow 1998].

4.2.1. Deliberative Architecture

Deliberative agents assume an explicit symbol model of their environment and capability of logical reasoning as a basis for intelligent actions based upon pattern matching and symbolic manipulation. This architecture has had little success in practice because of difficulty in manipulation of symbol structure [Jennings and Wooldrige 1995]. Within the scope of deliberative architectures and symbolic AI there has been the desire to allow agents to decide upon their own course of action in order to achieve their goals (Figure 4-2). It has also allowed more sophisticated agents to be developed that allow agents to find a sequence of actions which results in the achievement of their goal and has led to the implementation of agent, beliefs, intentions and desires (BDI) [Schroeder and Mora 1996].

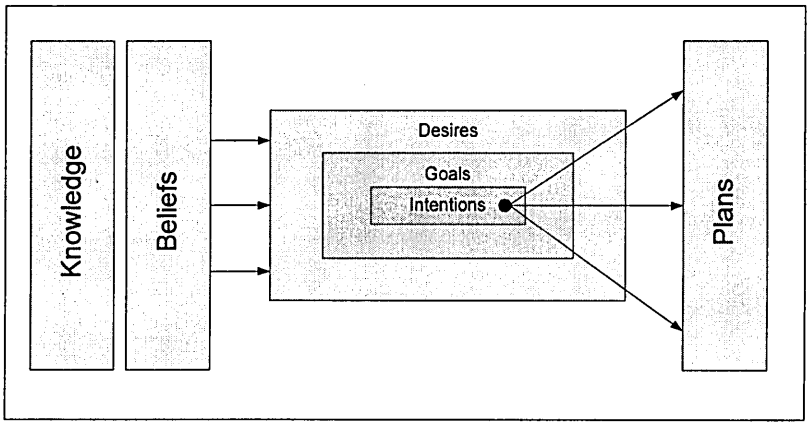


Figure 4-2: BDI structure

4.2.2. Reactive Architecture

Reactive architecture is an enhancement of deliberative architecture intended to overcome unsolved problems associated with symbolic AI. These problems have led some researchers to question the viability of the whole paradigm, and to the development of what are generally known as reactive architectures. A reactive architecture is defined as one that does not include any kind of central symbolic world model, and does not use complex symbolic reasoning. The agent does not necessarily need a complex structure to be able to act within a complex environment (Figure 4-3) [Brooks 1986]. It suffices to observe the environment precisely and recognise a range of simple principles. It does not need to create plans for acting, although plans can be used to optimise an agent's behaviour. For example, an agent often possesses incomplete information on its environment task and this is the cause of working in a dynamic environment, similar to a neural network architecture.

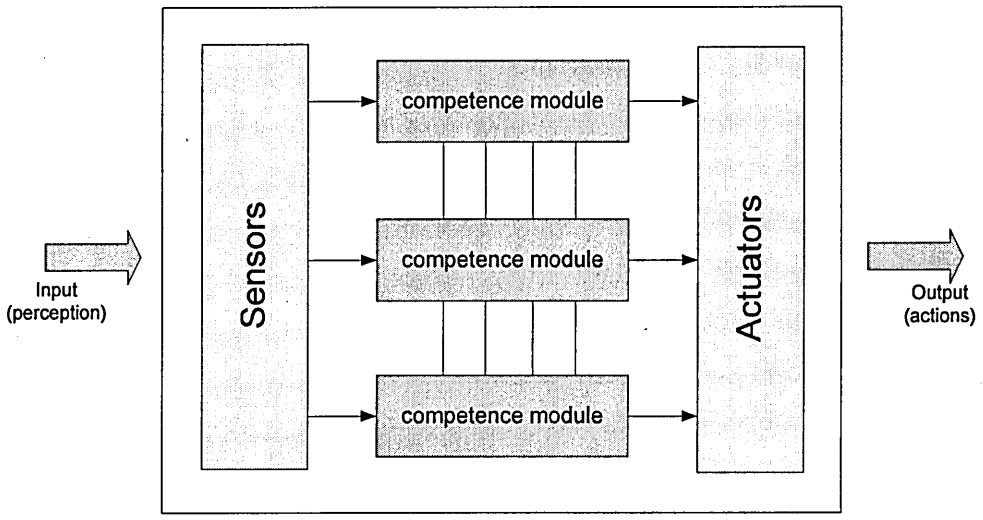


Figure 4-3: Architecture of reactive agents

4.2.3. Hybrid Architecture

Many researchers have suggested that neither a completely deliberative nor a completely reactive approach is suitable for building agents (Figure 4-4) [Brenner et al., 1998]. They have argued the case for hybrid systems, which attempt to combine classical and alternative approaches. The agent is able to differentiate between low level routine tasks in a reactive manner and to apply its powerful reasoning abilities for higher level and more advanced tasks [Wooldridge and Jenning 1995]. Based on the agent properties, there are several types of hybrid architectures for agent system solutions. For the purpose of this thesis, the focus will be on these three architectures.

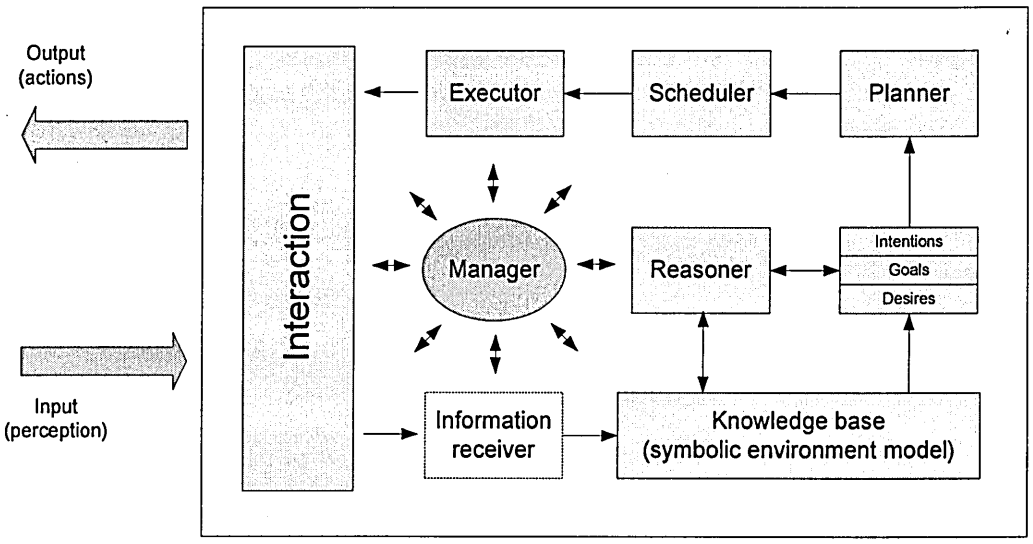


Figure 4-4: Architecture of hybrid agent

4.3. Multi-Agent Systems

In the previous sections, the discussion was focused on defining agents, and on classification and types of architectures. The properties of agents described in the earlier sections can be utilised for a multi-agent system approach. Presenting a specific agent's properties *per se* does not signify utilisation of the agent paradigm and technology. For example, in a single agent system, the one agent has to execute all tasks. Single agent systems are centralised and have several limitations. The agent is overloaded with large amounts of information and processing functions, lacks robustness due to single point of failure and the system is difficult to design, debug and maintain. A multi-agent system (MAS) is a computational system composed of several agents capable of interaction within the environment [Kraus 1997]. There are several advantages of the MAS approach. Firstly, it is able to present most of the agent properties. Secondly, it presents the criterion of decentralisation, which can be exploited by several types of applications inherited from distributed systems: functionally distributed, distributed decision-making and distributed reasoning. The distribution principle provides special tool of modular techniques that make the designing and implementation process much easier. In addition, such a technique also facilitates debugging and maintaining the system [Weiss 1999]. This is because the use of MAS makes it possible to break down the large complex

system at high level design, as independent agents that cooperate with each others to fulfil the overall goal of the system, and at a lower to focus on designing each agent. In other words, a MAS is logically decomposed into several simpler tasks which are distributed among a set of agents. Thirdly, MAS offers the following advantages [Hare and Jenning 1996]:

- Faster problem solving by exploiting parallelism.
- Decreased communication by transmitting only high level partial solutions to other agents rather than raw data.
- More flexibility by having agents with different abilities dynamically team up to solve a current problem.
- Increased reliability and fault tolerance by allowing agents to take on responsibilities of agents that fail.

In spite of several advantages of MAS, there are several issues that must be addressed in order for MAS to achieve their functionality. These include, task decomposition and allocation, coordination, cooperation, communication and negotiations [Brenner and Zarnekow 1998].

4.4. Adaptive Agents

An adaptive agent is a type of agent that is provided with adaptation of behaviour. Adaptation is the ability of an agent to operate in a dynamic environment. To cope with continuous change in the environment, the agent is provided with an ability to adapt to changing demands and opportunities. In such a situation, an adaptive agent is able to improve over time, based on experience of its environment. An adaptive agent can operate in a dynamic environment and react to unexpected events by generalising what it has learned during a training stage [Sen 1995,1996]. The ability to learn in MAS can avoid or reduce some problems, such as, which agents will be available at the time of emergence of a given issue, and how the available agent will have to interact in response to requirements [Familiar 1997]. We can simplify the idea by arguing that adaptation is similar to human's adaptive behaviour. Human have intelligence to adapt to any change of environment. They easily learn thousands of events that change in

their lives and when they face a similar event another time, they will react according to past experience, according to their goal or intention or belief [Imam and Kodratoff 1997].

Learning behaviour is the core element of an adaptive agent. Building an adaptive system that will start from a not-so-successful system into one that can achieve its goals, is often considered a better approach than building a successful system that does not change when the environment or goal changes [Berenji and Vengerov 2000]. Adaptive agents must be able to learn in order to improve their performance and adapt to changes. There are two approaches for using learning systems to build intelligent adaptive agents. The first approach is to build a system (agent) that acts as an intermediary between the learning system and the environment. This approach is more toward providing an intelligent user interface. The second approach is to modify the learning system to include temporal changes, sensory input, and constraint evaluation, in the learning function. In order to design an adaptive agent, the designer needs to decide not only what an agent should learn from the environment, but also how the agent will learn from the environment. There are several learning methods, such as, reinforcement learning [Sutton and Barto 1998, Maclin and Shavlik 1996] classifier systems model builders [Brenner et al., 1998] and reasoning [Lau and Hofestede 2001].

4.5. Agents in Information Retrieval System: A Review

The previous Chapter discussed the limitations of existing information retrieval (IR) approaches. In this section, we show how agent techniques have been applied to IRS and IFS. As stated before, the use of agents can reduce task complexity. Conventionally, complex tasks can be delegated among agents, which cooperate to fulfil the overall system goal. For example, one agent may be assigned to index documents, maybe using linguistic models, while another agent is responsible for presenting user interests, and other agents handle the vast number of different users, and so on. The complexity of the system is reduced by using agents, although the objective of the system remains the same.

Presently, already a number of IFS and IRS use agents' approaches. WebCrawler (www.webcrawler.com) uses several agents to independently provide search results,

which a search administrator acquires and indexes for storage in databases. Other tools using the same approach are MetaGer [meta.rrzn.uni-hannover.de], SavvySearch [guaraldi.cs.colostate.edu:2000/form] and MetaCrawler [metacrawler.com]. BargainBot [Bargain 1999] is an electronic shopping agent, which searches for specific products on the Internet. The system uses multiple connections to search online shops simultaneously. Users are given the comparative details of products found from which they can then make a decision. BargainBot is coded in PERL and is based upon a Multi-agent architecture, which allows the system to tolerate problems, such as non-response or erroneous returns.

JASPER (Joint Access to Stored Pages with Easy Retrieval), is an information agent system, which retrieves, summarises and stores information found on the WWW. *JASPER* uses ConText (a natural language system developed by Oracle) for tasks like extraction and summarisation. The agent learns about its user by keywords provided (by the user) and the WWW pages looked at (by extracting the keywords from these), and can update user profiles using so called, intelligent page store (IPS). IPS is also used to identify whether other users could be interested in the information.

InforSpiderSystem [Menczer and Belew 1994] uses an evolutionary algorithm to manage a *population of information agents*. A similar tool, *Amalthaea* uses a neural network for searching and learning, and genetic algorithms for the selection of agents [Moukas and Maes 1998]. It uses a multi-agent filtering and monitoring system using evolution algorithms for the selection and creation of new agents.

MACRON [Decker and Lesser 1995] is another multi-agent system based on the cooperative information gathering (CIG) paradigm. The system's architecture is based on partial global planning, using a centralised planner to generate sub-goals, which are taken up by other agents. It is designed for information gathering from heterogeneous sources, such as, newsgroups. Profusion [Gauch and Wang 1996], is an adaptive meta-search engine which analyses incoming queries, categorises them and automatically picks the best three search engines for the query based on *a prior* knowledge (confidence factor). It uses these confidence factors to merge the search result into a re-weighted list of the returned documents, removes duplicates and broken links and presents the final rank-ordered list to the user.

Fab [Balabanovic 1997, 1998] is an adaptive web page recommendation service. The task of text recommendation involves delivering sets of documents to the user based on the user model (the service seeks to adapt to its user model). These models are improved over time by giving feedback on delivered documents. In this system, both document and user are represented using the vector space model. The system function is comprised of three tasks:

1. Given one or more information source, pick a set of documents to present to the user, based on some user model (initially empty).
2. Obtain feedback from the user (either implicit or explicit).
3. Update the user model accordingly.

The Web Browser Intelligence toolkit (*WBI*) [Maglio 1997] is based on a model of what people do when they search for information on the web. The aim is to provide personal support for information searching and to effectively transfer knowledge gained by one person to another. First, behavioural data is collected from people searching for information on the web; second, the data is analysed to learn what the searchers were doing; and finally, a specific web agent is constructed to support such searching behaviour. To assist searchers, *WBI* has two personal web agents: the shortcut agent, to identify repeated search patterns and to suggest similar patterns for new searches (search relies on routine); and the way-point agent, to identify key nodes in finding a piece of information and maintain personal trails in terms of these (the search relies on location) [Maglio 1997].

The *NECI* meta-search engine [Lawrence and Giles 1998] improves the efficiency of web search downloading and analysing. Rather than waiting until all pages are downloaded and then displaying results that show the query term in context, pages are downloaded in parallel and the first result is typically displayed in less time than a standard search engine takes to display its response. *NECI* creates a summary for the users to find relevant documents faster and reduce the need for the user to access the full text of a document. It applies a uniform ranking measure to documents returned by different engines.

4.6. Agent Development Technologies

From the above discussion, it is evident that agents technology presents a new paradigm for design solutions. Agent technology, however, is not new and utilises existing technologies. The questions then are, how to develop an agents' system, what language to use; and the agent criteria. In order to utilise the characteristics of agents, most of the agent systems are being built with the following elements:

- Use of a specific programming language, such as, Java, C++ or Prolog.
- Use of communication language for communicating agents, such as, KQML or FIPA, ACL.
- Representation as strings, XML or any other object format such as GIF.
- Content language, such as, KIF or SL1

Developing agents from scratch using a specific language is difficult. The current practice in agent development is to use MAS environments, which integrate the agent technologies to fulfil the needs for development of agent systems, or as is commonly known agent frameworks or agent tools. Currently there are various agent frameworks have been developed such as JADE, ZUES and [Tools 2001]. The detailed issues on how to develop MAS will be discussed in Chapter five, while the actual implementation of the system is discussed in Chapter six.

4.7. Summary

In this chapter we presented a survey of relevant aspect of agent systems to show how in particular a multi-agent is appropriate paradigm to be used to design and implement systems. This Chapter has provided a review of agent-oriented approaches to system design and development. The fundamentals of agent technology are described through three examples of agent architectures: deliberative, reactive and hybrid architecture. The Chapter also discussed the main design issues in multi-agent system development. Finally, the Chapter discussed agent technology application in information retrieval systems. An overview of previous implementations was presented.

Chapter 5. PROPOSED INFORMATION RETRIEVAL SYSTEM DESIGN

5.1. Introduction

This Chapter discusses the development of the information retrieval system (IRS) developed in this research study. As discussed in agent technology that the traditional approach of developing an adaptive system is performed machine learning techniques to the system which deal with static data. The adaptive behaviour is performed according the user action rather performed independently. The proposed IRS presents a real adaptive behaviour when using agent technology. The requirement of IRS system as discussed in Chapter 2 clearly shows that the proposed IRS system deal with dynamic, large and distributed data, which very appropriates used agent solution. The IRS is a complex system and processes large volumes of dynamic and unstructured information which contains of various decomposition of complex processes such as downsizing unstructured information, indexing the document, two level of filtering, ranking, and applying machine learning. Based on discussions in Chapter 3 and 4, the use of an agent approach is the closest paradigm for adaptive IR system. Multi-Agent approach promises the following advantages:

- Faster processing time: utilising the parallel task execution, using agents, reduces the response time.
- Improved flexibility – The nature of multi-agent systems allows any number of agents to be created, whereby each agent can be implemented and executed independently to perform a specific functionality.
- Easy of maintenance – each agent is assigned a specific functionality as part of the overall task, and hence, maintenance can be carried for each agent without affecting other agents.

The first advantage is the benefits of having real adaptive system, while the rest is the beneficial of choosing MAS architecture solution. These advantages cannot be applied

using the traditional approach therefore an MAS paradigm is the best paradigm for the proposed system.

5.2. Adaptive IRS needs an MAS paradigm

The criteria of real adaptive IRS are presented while using the agent approach. In Chapter 4, discussed various IRS such as WebCrawler, JASPER, MACRON and Fab which have been developed using agent approach to present MAS criteria. However, the proposed of IRS are different from others because we are applying a real MAS approach. Those systems using MAS approach by creating many agents to perform similar task to increase performance of the system. However the proposed IRS applies real MAS, which the system is divided into various sub-processes that performs in parallel. The use of MAS approach allows each sub-process to be free to perform in its own effort without waiting other sub-processes. This approach is suitable approach of having many system users, as we know an IRS could be used by thousand of user at the same time for example the total number of Internet user now is more than a billions.

The concept is similar to the transmission of network packet. Each sub-process is like a packet which provides with address and owner of the sub-process. However the decision of splitting the sub-process is done by system designer, rather automated like packet. This approach provides an environment that all independent sub-processes can be performed in parallel and dependent sub-processes will negotiate among them. In the following sub-section will discuss more detail how MAS approach increases the performance of the system (reducing time).

Due to the complexity of the system and the fast growing of information in the Internet that causes the reduction in performance of an IR system, MAS approach is considered as a suitable paradigm for designing and developing an IRS .

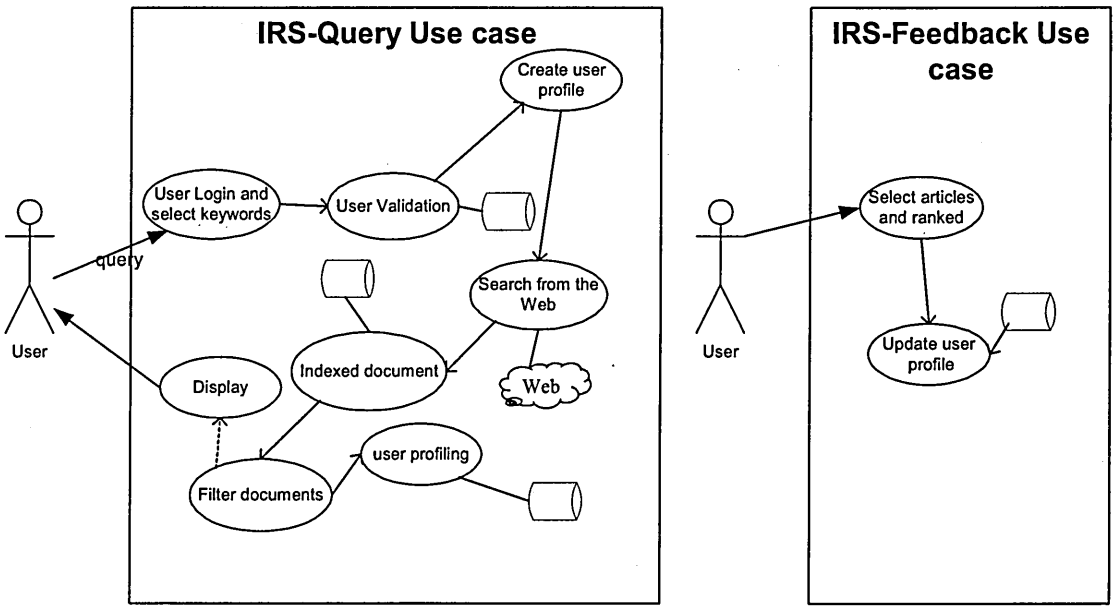
Although increasing hardware processing capability is another solution, but looking at designing perspective using MAS approach is believed as part of technique to maintain the performance of the system while dramatically increase information in the internet [Shelly, et. al, 2003].

5.3. Development Life Cycle for Proposed System

We provide the complete development of each agent through the first two agents namely search agent and user interface agent in here to illustrate the process and notation for design. We then focus on the three key agents namely, user model agent, document agent and filter agent that we have specifically designed for this proposed system. The first because of its importance in the way we have chosen to index documents for effective IRS. It aids the specialisation and exploration aspect of personalisation as identified in our introduction chapter. The second because its details are appropriate to the adaptive aspect of personalisation in that it is novel in combining of GA, relevance feedback and fuzzy logic.

5.3.1. Functional Requirements Analysis

The proposed IR system has been developed on a multi-agent paradigm to represent the different typical activities, including document representation, query formulation, user-needs modeling, filtering and user-needs model reinforcement. The system requirements are represented using Use Case models as shown in Figure 5-1 (a, b).



a. Event Use case: User enquiry *b.Event Use case: User Feedback*

Figure 5-1:The IRS Use Cases

The first Use Case shows the activities when the user makes an enquiry . The Use Case shows two objectives: to assign keywords of interest and provide feedback on the relevance of the ranked documents, that build upon the user model. The second Use Case shows the activity of the user providing relevance feedback. The most accurate results are obtained when the frequency of enquiry and feedback is high. Therefore, the more feedback provided, the more accurate the user profile and the more accurate result.

The proposed system is designed to act as a personalisable and adaptable tool for information retrieval from the Internet. In this regard, the "black box" view of the system consists of a document collection from the Internet as input, while the output is the processed documents organised according to user interests and preferences. In order to achieve the system goal, the user needs to communicate with the system.

The Use Cases above is combined to produce overall the Use Case for the IRS, as shown in Figure 5-2:

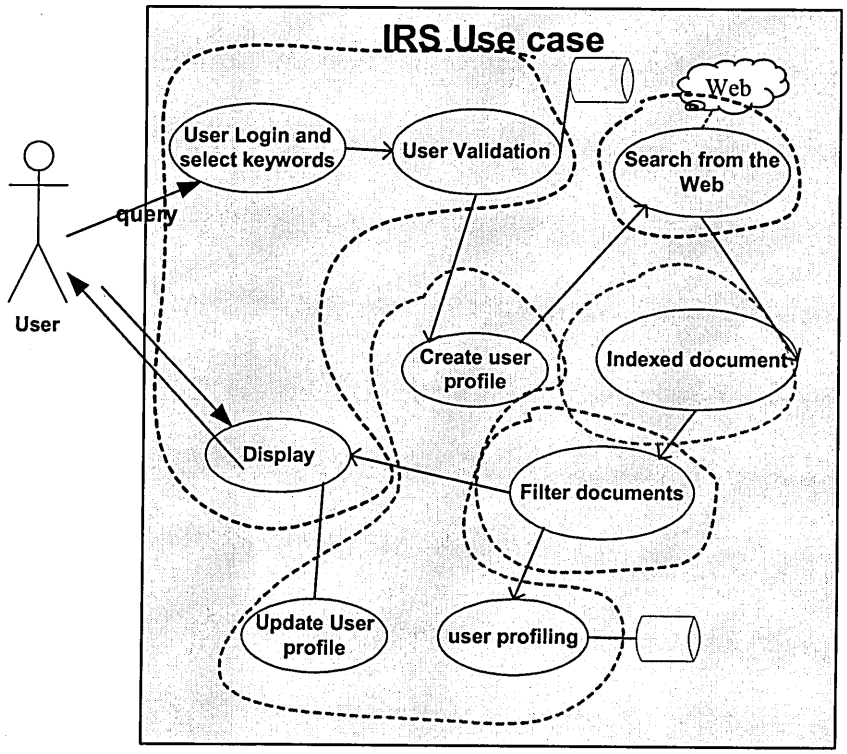


Figure 5-2: The IRS Use Case

5.3.2. System Level Design

The purpose of agent level system design is to identify agents and their interactions. There are two types of models that illustrate agents and their interaction. The first model is called agent system architecture, which shows the static diagram of the agents and their interactions. The agent system architecture portrays the overall IR agent system. The second model describes the dynamic modelling showing time-based agent interaction, called the system plan. Most agent-oriented methodologies adopt the first type of model, although they may use different model notation. System plan modelling uses unified modelling language (UML) [Odell et al., 2000, Booch 1999].

5.3.2.1. Agent System Architecture

The analysis of the Use Cases as shown in Figure 5-3, groups those Use Cases into five independent task categories, which are assigned to agents. Thus, five agents are identified to represent the IRS solution: user interface agent, search agent, document agent, filter agent and user model agent. The agent system architecture is shown in Figure 5-3.

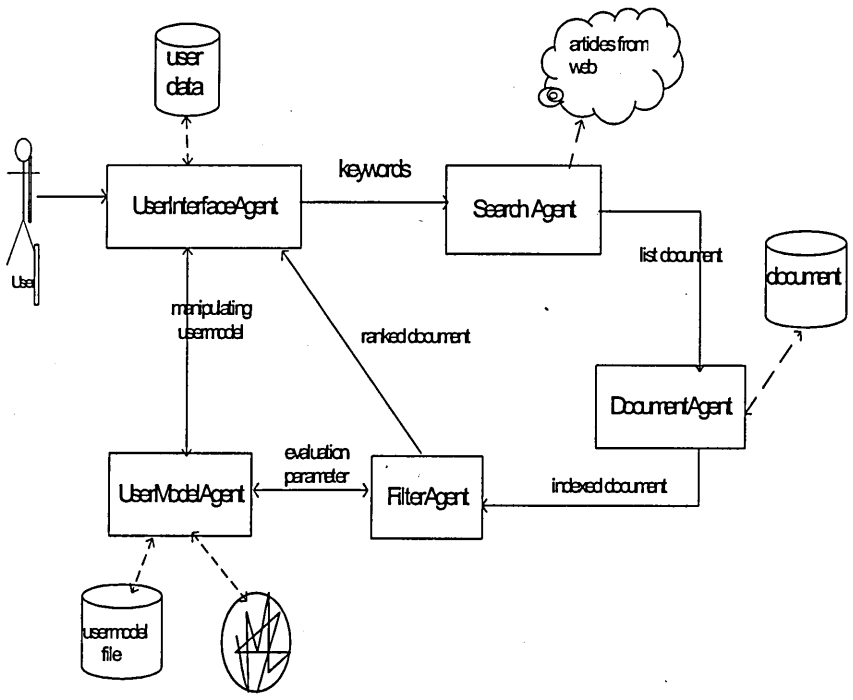


Figure 5-3: Overall IR System Architecture

The "box" symbol describes an agent, the "can" denotes a resource, the "cloud" a dynamic information source, the "ellipse" illustrates the "intelligence" in the system. The "dashed line" connections describe agents located in same location, while the "solid line" illustrates agents located in different locations.

The *user interface agent* is designed for interaction with humans. It creates *n* number of user interfaces for *n* number of users. The functions of the user interface consist of user login, query keywords input, display result and provide feedback.

The *search agent* acts as a meta-search tool of any Internet search engines. It uses the keywords to retrieve documents, which are then passed to the *document agent*. The *document agent* indexes the documents using normalised keywords. As a result, the highest indexed documents are sent to the *filter agent*. In order to increase the precision, the *filter agent* ranks the indexed document, according to the user model. Therefore, two levels of filtering are performed, through indexing and ranking processes. However the ranking function provides more precise filtering according to representation of the current model of user information needs. The ranked documents are then shown to the user through the *user interface agent*. This allows the user to evaluate the relevance of the ranked documents, by giving a score to each document in the form of fuzzy values. This user-feedback modifies the representation of user-needs. In order to affect a perpetually evolving user model, the *user model agent* maintains a population of competing models, which evolve using genetic algorithm.

5.3.2.2. System Plan Model

The IRS system plan model is shown in Figure 5-4. The system plan shows two roles of the user interface agent: make query and feedback. The search agent, the document agent, filter agent and user model agent provide their services autonomously. The modeling notations presented in the figure follows AUML [Odell et al., 2000]. In order to achieve the system goal, the user communicates with the system, via an interface with two objectives: (1) provide interest keywords and (2) provide relevance feedback on the retrieved documents. The system automatically creates new user profiles most relevant for an existing user. The retrieved information is then indexed to become a concrete document, based on the user profiles. The result is displayed to the user, so that the user can provide feedback on the articles.

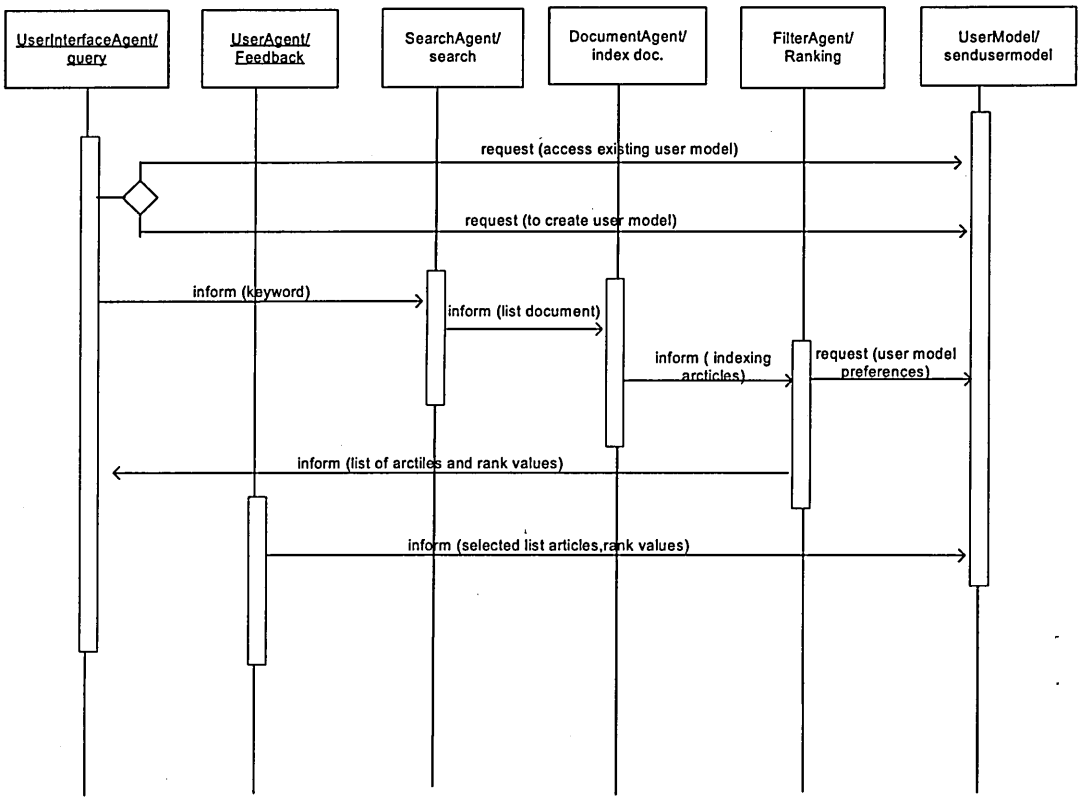


Figure 5-4: IR System Plan.

5.3.3. Agent Level Design

An agent collaboration diagram was used for modelling at the agent level. CRC cards are an object-oriented modelling technique used to represent object collaborations. CRC cards are part of a development approach called Responsibility Driven Design (RDD). The aim of RDD is to look at the overall functionality of the system and divide this functionality up between classes [Wirfs-Brock and Wilkerson 1990]. CRC cards provide an effective technique for exploring possible ways of allocating responsibilities (high level description of function to classes and the collaborations that are necessary to fulfil the responsibilities) [McRobb 1999].

5.3.3.1. User Interface Agent

This agent is responsible for meditating communication between the external user and the rest of the system (other agents). The User interface agent collaboration model is shown in Figure 5-5.

User Interface Agent Collaboration Model

User Interface Agent CRC Card	
Responsibilities	**Collaborations**
Login to the system	
Form the query	Search Agent
Display result	Filter Agent
User evaluation (Feedback)	User-model Agent

Figure 5-5: User Interface Agent CRC Card

Responsibilities

All the interaction with the user is performed through a dialogue box. The following is a short description of each responsibility.

- Login to the system is the first thing that any user should perform, whether a new user or an existing user.
- The query (keywords) is received from the user and enhanced by suggesting additional keywords to the user.
- The agent is also responsible for transferring the query to the search agent.

- The results are displayed to the user in form of a ranked list.
- The user is able to evaluate the results, and transfer the feedback to the learning agent in order to update the user profile.

Collaboration

The User Interface Agent communicates with Search Agent, Filter agent and User-model agent. The collaborations are shown in the figure below, in terms of recipients and sources of messages.

Agent Name: User Interface Agent	
Sending message to	***Receiving message from***
Search agent: **Message type:** *Request* **Content** : *do search for query*	**Filter agent:** **Message type:** *Inform* **Content** : *ranked document result*
User-model agent: **Message type:** *Inform* **Content** : *user name or new user*	
User-model agent **Message type:** *Request* **Content:** *Update user profile by using score value.*	

Figure 5-6: User Interface Agent Collaboration

5.3.3.2. Search Agent

The search agent performs the task of submitting queries in the correct manner, and gathering the information from the source (Internet).

Search Agent CRC Card	
Responsibilities	**Collaborations**
Perform search	User Interface agent Document Agent

Figure 5-7: Search Agent CRC Card

Responsibilities

In order to perform the search, existing search engines are used because they are efficient in terms of high recall and fast response time. The results of this process are then transferred to the document agent. The overall goal of this research is to develop an IR system which uses the World Wide Web as a source of information. Any commercial search engine such as Google can be used to retrieve text from the Web. Google and many other search engines provide APIs that allow a programmer to interface with their content, and retrieve the data in a more convenient form. Thereafter we will process that data to find sets of relevant document. However in this research BIDS have been used because the proposed IRS is text based only.

Collaboration

This agent communicates with the Document Agent and User Interface Agent. The collaboration is shown below as a table of recipients and source of messages.

Agent Name : Search Agent	
Sending message to	***Receiving message from***
Document agent: **Message type:** *Inform* **Content** : *search result*	**Search agent:** **Message type:** *Request* **Content** : *do search for query*

Figure 5-8: Search Agent Collaboration

5.3.3.3. Document Agent

The goal of this agent is to create a representation of each document, in other words, indexing the documents.

Document Agent CRC Card	
Responsibilities	**Collaborations**
Create document index (represent document content)	Search agent Filter agent

Figure 5-9: Document Agent CRC Card

Responsibilities

The representation typically used in many systems is a set of features derived from the document collection. There are many features of a document that could be useful to capture in a representation language. For instance, visual appeal of an intended audience, complexity of writing, quality of writing, genre or style and the subject of

matter. In this thesis, we represent only the subject of document; each document in a collection is represented as a list of subject keywords. However, before becoming features these words will typically have the case of their letters normalized, and certain types of words, such as, prepositions, determiners, and pronouns removed from the feature set.

There exists a large class of words that have no inherent meaning when taken out of the context and, thus, are basically useless as index terms. For example, terms such as 'the', 'and', 'or', 'of' have no semantic content. These words also tend to be among the most frequently occurring terms in English language. Therefore, it seems natural to filter these terms out of index terms. This is done by creating a list of terms that not to be indexed, otherwise known as a stop list. Using a stop list of function words can significantly improve the efficiency of a retrieval system by reducing both the size of index terms and the time required to conduct a search. Another common strategy for reducing index size and potentially improving retrieval performance is to create word equivalence classes for index terms by removing and modifying prefixes and suffixes to identify the root form of the word. This is achieved by applying a stemming algorithm to word tokens. A number of different stemming algorithms have been proposed. Most stemmers work by iteratively applying a set of rules for prefix and suffix removal. A simple rule of this form is one, which converts the plural to the singular form.

An additional subtlety to the representation of documents is the assignment of a numerical weight to all terms in a document collection. The weight assigned to a term occurring in a given document is an attempt to quantify that term's importance to the subject of that document. There are many methods for calculating the weight of a term. Most are statistical, based on the term's frequency of occurrence within a collection, known as the inverse document frequency (idf), and on its frequency of occurrence within the document, known as the term frequency (tf). The term weighting function, shown in equation 5-1, is typical of such methods. The weight resulting from this function is often referred to as a tf•idf weight . Let $D = \{d^1, d^2,.....,d^n\}$ denote a set of documents and $T=\{t^1, t^2,.....,t^n\}$ a set of all terms of the documents. The weight d^j_{ti} of a word t_i in document dj is given by,

$$d_{t_i}^{j} = (0.5 + 0.5\frac{tf_i}{tf_{\max}})(\log\frac{n}{df_i}) \qquad (5\text{-}1)$$

Where tf_i is the number of times word t_i appears in document d^j (the *term frequency*), df_i is the number of documents in corpus which contain t_i (*the document frequency*), n is the number of documents in the corpus and tf_{max} is the maximum term frequency over all words in d^j.

Collaboration

This agent has communication with search agent and filter agent. The collaborations are shown below as a table of recipients and source of messages.

Agent Name: Document Agent	
Sending message to	***Receiving message from***
Filter agent: **Message type:** *Inform* **Content** : *Vector of indexed document*	**Search agent:** **Message type:** *Inform* **Content** : *search result*

Figure 5-10: Document Agent Collaboration

5.3.3.4. User-model Agent

The remit of this agent is to guide the user in the query formulation process and to store and manage the user's interest in the form of a user profile. This agent is a key element of the system architecture and incorporates genetic algorithms, relevance feedback and fuzzy logic.

User-model Agent CRC Card	
Responsibilities	**Collaborations**
Create initial user profile	User Interface agent
Update User profile	User Interface agent
Finding the best query	Filter Agent

Figure 5-11: User-model Agent CRC Card

Responsibilities

This agent is responsible for creation and update of the user profiles. Part of the functionality of the agent system is building an initial user profile, which is stored as a set of weighted keywords. In order to do that, the system exhibits collaboration behaviour, by allowing user to build profiles by sharing other profiles in a similar search area. The main functionality of this agent is to adjust the representation of the user's interest so it is consistent with dynamic user feedback.

As the volume and variety of available information continues to grow, it is increasingly difficult to obtain information that accurately matches user needs. This is due, firstly, to the fact that users often do not present to information retrieval systems queries that optimally represent the information they want, and secondly, the measure of a document's relevance is highly subjective and variable between different users. This research addressed this problem by proposing an approach that relies on evolutionary user-modelling in order to retrieve domain-specific information. It describes adaptation in the proposed information retrieval system, which learns user needs from user-provided relevance feedback. The method combines qualitative feedback measures using fuzzy inference, and quantitative feedback using genetic algorithms (GA) fitness measures. The user model-agent is responsible for adaptive behaviour of the system.

In the user-model agent, a GA [Goldberg 1989] is used to evolve and adapt query vectors, which are models of user information needs. With GA, user models represent hypothetical knowledge about the user needs, encoded in a chromosome. These chromosomes are expressed in keyword terms and their weights, unlike in previous studies [Vrajitoru 1998] where only terms were used. Each chromosome, thus, is a hypothesis on how to evaluate the relevance of a document, and competes against other chromosomes to predict user satisfaction from the retrieved documents. It is assumed that the users' information needs are stochastic, but non-transient. In other words, the information needs vary in a non-deterministic manner between users, but they do not change rapidly over time. Or, it can be said that while the perceived relevance of the same documents may vary widely between different users, each user's perception of relevance of the documents does not change rapidly over time. In this regard, the GA can be used to evolve a user model for one user (or specific group of user). The GA through user feedback, however, can effect an adaptation of user needs to new areas, thus, improving and maintaining the retrieval precision in real-time.

GA is blind search mechanisms; hence for online learning it is advantageous to initialize the population with any *a priori* knowledge in order to avoid undesirable performance in the early generations. In this research, the initial chromosomes comprise terms from a vocabulary that was judged relevant for individual users, and the assignment of fitness for the initial population is proportional to the mean similarity between all chromosomes in the population set, given by,

$$f_0(q_i) = \frac{1}{n}\sum_{j=1}^{n} sim(q_i, q_j) \qquad (5\text{-}2)$$

Where n is the population size. This kind of initialization also helps people who have problems in finding the right keyword, but can distinguish relevant from irrelevant documents; the system builds up the proper query for them.

A key role of GA in user-needs modeling is to continuously modify the representation of user needs, based on a quantitative and a qualitative relevance metric. The quantitative metric is proportional to the mean similarity between the query vector and all (or ranked) retrieved documents. The qualitative metric relies on a fuzzy relevance feedback from the user. Thus, the user provides a linguistic value of feedback for a retrieval, which is used in a fuzzy inference system to obtain a crisp feedback measure.

The crisp feedback is combined with the measure of similarity between the query vector and the retrieved document to determine a quantitative metric used to adjust the fitness of the chromosomes in the population. This is a novel learning approach known as Interactive Evolutionary Computation [Hideyuki 2001].

We also defined a new two-point crossover technique. Firstly, we generate two random gene positions to be applied as crossover positions. Then, we copy all genes from the first parent to the same position in the offspring chromosome (child) up to the first crossover site. This procedure is repeated for the genes after the second crossover position. For the genes between two crossover positions, we exchange the genes of the first parent with those of the second parent in the same position. To avoid repeated terms in new offspring we select randomly another term and its weight from a pool of terms extracted from documents. This is illustrated in Figure 5-12.

Mutation is carried out as follows: for each chromosome in the population, and for each gene within the chromosome, we generate a random number r from range of (0...1). If r $<p_m$ the gene will be mutated by replacing it with another gene from a pool of terms, otherwise the gene is unchanged.

$$child1(i) = \begin{cases} parent1(i) & \text{if } i < crosssite1 > crosssite2 \\ parent2(i) & \text{if } crosssite1 \le i \le crosssite2 \text{ and } parent2(i) \notin child1 \\ select\ randomly\ from\ pool\ of\ terms & \text{if } crosssite1 \le i \le crosssite2 \text{ and } parent2(i) \in child1 \end{cases}$$

$$child2(i) = \begin{cases} parent2(i) & \text{if } i < crosssite1 > crosssite2 \\ parent1(i) & \text{if } crosssite1 \le i \le crosssite2 \text{ and } parent1(i) \notin child2 \\ select\ randomly\ from\ pool\ of\ terms & \text{if } crosssite1 \le i \le crosssite2 \text{ and } parent1(i) \in child2 \end{cases}$$

Figure 5-12: Crossover operator

5.3.3.4.1 GA Operators: Examples

Two query vectors with n keywords (terms) Query1 and Query1 can be represented as

$Query1 = (q_1term_1, q_1term_2, q_1term_3 .... q_1term_n)$

$Query2 = (q_2term_1, q_2term_2, q_2term_3, ...q_2term_n)$

We use a weighted term vector, so that each $term_i$ (or $qterm_j$) contains keywords and weight ranges between 0 and 1.

Query1 = {(Artificial Intelligence,0.2), (Internet,0.96), (Natural Language Processing,0.1), (Indexing,1.0) , (Information Retrieval System, 0.99), }

Query2 = { (Expert System, 0.88), (Artificial Intelligence,0.76),(Expert System,0.5), (Neural Network,0.3), (Adaptive System, 0.69) }

pool of terms = { Semantic network , Fuzzy system, Information Filtering, Knowledge base, Precision, Ranking, Search Engine,.....}

If *crosspint1* = 2 and *crosspoint2* =3 then in by using standard 2-point Crossover the new children will be:

between 0 and 1.

child1 = {(Artificial Intelligence,0.2), (Artificial Intelligence,0.76), (Stemming Algorithms 0.5), (Indexing,1.0) , (Information Retrieval System, 0.99), }

child2 = { (Expert System, 0.88), (Internet,0.96),(Natural Language Processing ,0.1), (Neural Network,0.3), (Adaptive System, 0.69) }

Looking at new query (*child1*) we can see that Artificial Intelligence has appeared twice in the term list, one of them with high value weight, which means that the user is interested in documents relevant to Artificial Intelligence. However the other one with low weight shows that the user is not interested in the subject. As the result shows a conflict; the solution is to apply the new defined crossover, which will use the existing pool of terms e.g. randomly first element of the list is selected (Semantic Networks). The new children will be:

child1 = {(Artificial Intelligence,0.2), (Semantic Networks ,0.87), (Stemming Algorithms, 0.5), (Indexing,1.0) , (Information Retrieval System, 0.99), }

child2 = { (Expert System, 0.88), (Internet,0.96),(Natural Language Processing ,0.1), (Neural Network,0.3), (Adaptive System, 0.69) }

Looking at the new query e.g. *child1* means that the user is looking for any document relevant to an application of stemming algorithms and semantic networks in information retrieval; the same rule applies for mutation.

5.3.3.4.2 Fuzzy Relevance Feedback

Fuzzy logic [Zadeh 1965] has been applied in this research because it provides a very convenient methodology for "computing with words" (linguistic as opposed to numerical values). Although words appear less precise than numbers, their use in information processing is closer to the human perception of concepts. A fuzzy inference system (FIS) is a rule-based system that associates a set of inputs (conditions) with a set of rules, to obtain an output.

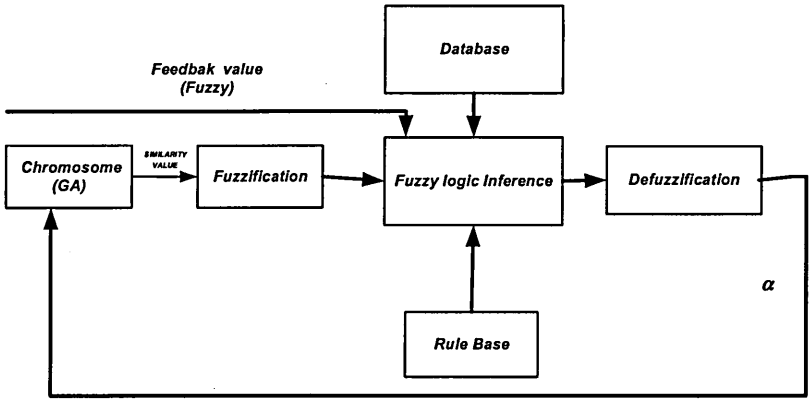


Figure 5-13: Fuzzy Inference System

The rules for the FIS are shown in 5-13. Each of the cells in the Table represents an IF <conditions> THEN <action> statement. For example, the first cell in Table 5-1 is a statement: IF<Similarity = P; Feedback = E> THEN <Adjustment = IH>, where P, E, IH stand for the fuzzy linguistic values "Poor", "Excellent" and "Increase High", respectively. These rules, in general, are heuristic and rely on obtaining knowledge from

a domain expert. In the fuzzy system, the universes of discourse over which these linguistic values are defined are **similarity** ∈ (0.0 ... 1.0) and **feedback** ∈ (0.0 ... 1.0) and **adjustment** ∈ (-0.4 ... 0.4}. An example of the specification of the linguistic values, for **similarity**, is shown in Figure 5-14. Detail of the operation of FIS is provided by [Lee 1990].

Feedback

Similarity	E	VG	G	M	P	VP
P	IH	IH	IM	IL	DL	DM
M	IH	IH	IM	IL	DM	DM
G	IM	IL	IL	DL	DL	DH
VG	IL	IL	DL	DM	DH	DH

Table 5-1: FAM Table for User Model Adjustment

Similarity Values: Poor (P), Moderate (M), Good (G), Very Good (VG)

Feedback Values: Excellent (E), Very Good (VG), Good (G), Moderate (M), Poor (P), Very Poor

Output Values: Increase High (IH), Increase Moderate (IM), Increase Low (IL), Decrease Low (DL),.

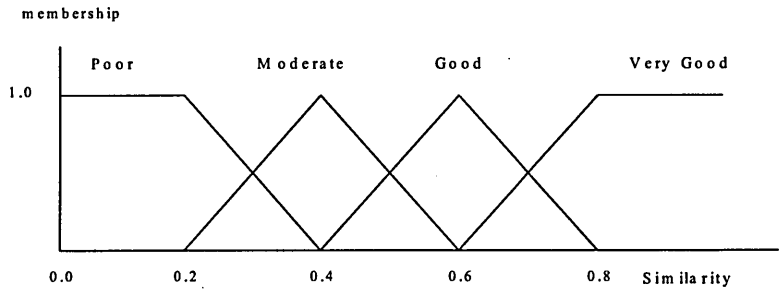


Figure 5-14: Specification of linguistic values

The output of the inference system is used to adjust the fitness of the chromosomes competing to evolve the optimal user-needs models. However, since only one chromosome is, in fact, used to retrieve documents, and thus evaluated, the fitness of the rest of the population is adjusted in proportion to their distance from the evaluated chromosome. This is given by,

$$f'(q_j)=(1-\beta)f(q_j)+\left[\alpha+\frac{1}{n}\sum_{i=1}^{n}Sim(Q_0,d_j)\right]f(q_j)sim(Q_0,q_j) \qquad j=1\cdots n \tag{5-3}$$

Where *f(q)* is the old fitness, β is a fitness sensitivity factor that limits the change and maximum value of fitness, Q_0 is the evaluated (query) chromosome, q_j are the rest of the chromosomes in the population, d_j are the retrieved documents and α is determined from the fuzzy inference system. The output of the FIS is also used to adjust the term weights in the chromosomes. This is due to the observation that the initial term weights, as calculated using term frequencies, may not in fact be an accurate representation of importance of the documents to the user. Hence, we need to update with actual user relevance feedback. The term weight adjustments, on the other hand, are given by,

$$w'(q_j) = w(q_j) + \alpha w(d_0) \quad \forall\, t_j = t_0 \tag{5-4}$$

where d_0 is the retrieved document vector, q_j are the query chromosomes and t_i the terms in the query and document vectors. The resulting effect is that, for those terms already present in the user model, term weights will be modified by the feedback and terms not

already present in the model may be added to it. Terms are also removed from the model representation when their weights are zero or less, and the total number of terms is limited by the predefined chromosome size.

Collaboration

This agent has communication with Filter Agent and User Interface Agent. The collaborations are shown below as a table of recipients and sources of messages.

Agent Name : User-model Agent	
Sending message to	***Receiving message from***
Filter agent: **Message type:** *Inform* **Content** : *Best chromosome vector*	**Filter agent:** **Message type:** *Request* **Content** : *find fittest chromosome*
	User Interface agent: **Message type:** *Inform* **Content** : *user login name or new user*
	User Interface agent: **Message type:** *Inform* **Content** : *user name or new user*
	User-model agent **Message type:** *Request* **Content:** *Update user profile by using score value.*

Figure 5-15: User-model Agent Collaboration

5.3.3.5. Filter Agent

Traditionally, search agents retrieve a set of potentially relevant documents from the Internet. This retrieval is efficient in terms of high recall rate and fast response time, but at the cost of poor precision. Recall rate is the percentage of documents that are retrieved, while precision is the percentage of documents retrieved that are considered relevant. In this research, the Filter Agent improves the precision by ranking documents according to user preferences, using a user model.

Filter Agent CRC Card	
Responsibilities	**Collaborations**
Rank the result (filter document)	Document Agent User-model Agent User Interface Agent

Figure 5-16: Filter Agent CRC Card

Responsibilities

Improving the precision of retrieved documents is possible, dependent on an ability to measure accurately the similarity between a query vector (user-needs) and a document vector (retrieved documents). This, moreover, is complicated by the fact that the measure of the relevance of a document to a user is not objective, and therefore difficult to express quantitatively. We have proposed to overcome this by applying two methods to determine the similarity; a quantitative (objective) measure based on term weights, and a qualitative (subjective) measure based on user relevance feedback.

The *filter agent* attempts to rank documents according to user preferences, by measuring the similarity between a query vector and a document vector. To assign a numeric score to the relevance of a document a query, the model measures the similarity between the query vector (since query is also just text and can be converted into a vector) and the document vector. The similarity between two vectors is once again not inherent in the model. Typically, the angle between two vectors is used as a measure of divergence between the vectors, and cosine of the angle is used as the numeric

similarity. Let $\vec{D}$ and $\vec{Q}$ be two vectors that $|\vec{D}|$ is for magnitude of the document and $|\vec{Q}|$ is for magnitude of the query, θ is angle between the vectors as shown in Figure 5-17.

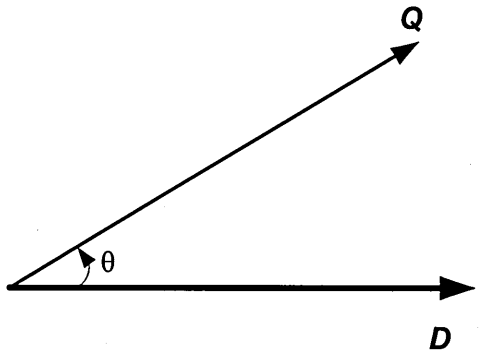


Figure 5-17: Dot product geometry

The dot product $\vec{D}.\vec{Q}$ Of the vectors is defined to be product of the magnitude of the vectors $\vec{D}$ and $\vec{Q}$ with the cosine of angle θ between the two vectors,

$$\vec{D}.\vec{Q} = DQ\cos\theta \qquad (5\text{-}5)$$

The θ can vary from 0 to π. Since cosine has the nice property that it is 1.0 for identical vectors and 0.0 for orthogonal vectors. As an alternative, the inner-product (or dot-product) between two vectors is often used as a similarity measure. Due to the fact the vectors are forced to be unit length, then the cosine of the angle between two vectors is same as their dot-product. The dot product of two vectors that are perpendicular to each other is zero since the angle between the vectors is π/2 and cos(π/2) = 0 and if θ = 0, cos(0) = 1 then the dot product of two vectors are bigger than zero and the maximum value is 1 if two vector are identical and normalized.

If $\vec{D}$ is the document vector and $\vec{Q}$ is the query vector, then the dot-product similarity

$$Sim\ (\vec{D},\vec{Q}) = \sum_{t_i \in Q,D} w_{t_{iQ}} . w_{t_{iD}} \qquad (5\text{-}6)$$

between document D and query Q can be represented as:

Where $w_{t_{iQ}}$ is the value of i^{th} component in the query vector $\vec{Q}$, and $w_{t_{iD}}$ is the value i^{th} component in the document $\vec{D}$. Since any word not present in either the query or the document has a value of 0, respectively, we can do the summation only over the terms common in the query and the document. Following equation shows normalized form of 5-6 equation in order to have a similarity value between 0 and 1.

$$Sim\,(\vec{D},\vec{Q}) = \frac{\sum_{i=1}^{n} w_{t_{iQ}} \cdot w_{t_{iD}}}{\sqrt{\sum_{i=1}^{n} (w_{t_{iQ}})^2 \sum_{i=1}^{n} (w_{t_{iD}})^2}} \qquad \frac{n}{\sqrt[n]{n}} \neq 1 \qquad (5\text{-}7)$$

The similarity value of 0 means there isn't any common term in the query and document, indicates two vectors are orthogonal. the value of 1 for similarity shows that all terms are common in the query and document also the term's weight are similar as well which indicates two vectors are coincide. Any value between zero and one shows some terms are common but may have different weight value.

Collaboration

This agent has communication with Document Agent, User-model Agent and User Interface agent. The collaborations are shown below as a table of recipients and source of messages.

Agent Name : Filter Agent	
Sending message to	***Receiving message from***
User Interface agent: **Message type:** *Inform* **Content** : *ranked document result*	**User-model agent:** **Message type:** *Inform* **Content** : *Best chromosome vector*
	Document agent: **Message type:** *Inform* **Content** : *Vector of indexed document*

Figure 5-18: Filter Agent Collaboration

5.3.4. Non Functional Requirement

The multi-agent system approach increases the performance of IRS compared to the traditional approach executing tasks in a sequence. In the multi-agent systems approach, each sub-task is assigned to an agent, which provides ability to perform tasks autonomously, in parallel. Thus, an obvious advantage of the multi-agent approach would be to reduce the response time to search queries. Figure 5-19 shows the general arrangement of activities performed in IRS. The figure shows *n* users make enquiries at the same time. The enquiries go into queue following a first-in first-out sequence.

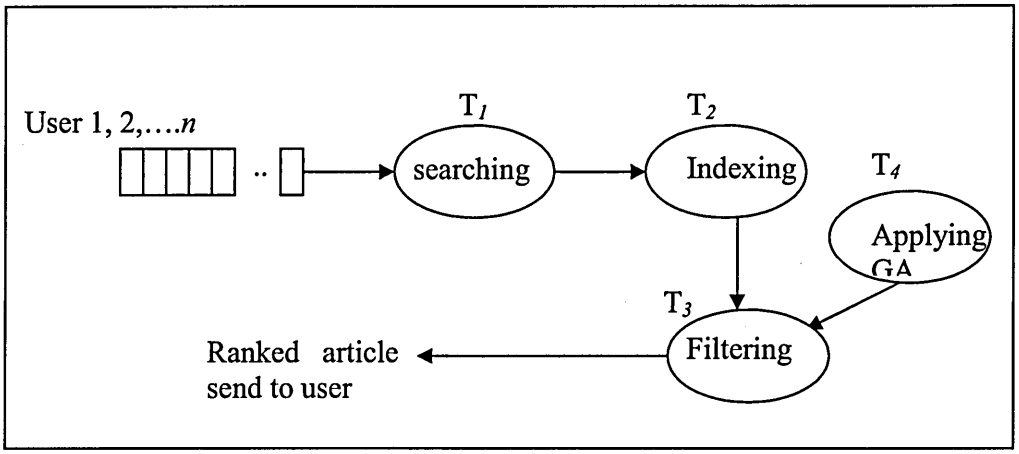


Figure 5-19: The sequence approach of task execution

Each enquiry from a user is comprised of, searching, indexing, and filtering and machine learning, as stated previously. It is obvious the user queue rapidly grows longer as n increases, causing a delay in the response time. A comparison of the traditional and multi-agent system approaches can be illustrated as follows:

Consider an IRS with k tasks, (T_1, T_2, T_3 and T_4). A multi-agent system comprised of k agents executed in parallel will respond k times faster a sequential system. This, however, is in general not achievable because of task dependencies. Let U be the total time to perform the sequence tasks, and lets p the mean number of agents that can be executed simultaneously in the multi-agent system. The total time to execute the k tasks using the sequential system is given by,

$$U = \sum_{j=1}^{k} t_j \tag{5-8}$$

The total time for multi-agent system, on the other hand, is given by,

$$U = \sum_{i=1}^{k} t_i - \sum_{j=1}^{p-1} t_j \tag{5-9}$$

If we make assumption that each agent executed in the same time then

$$U = (k - p + 1) * t_p \qquad (5\text{-}10)$$

Where m is the number of independent agents which can run in parallel. Hence agents' system is speeded up at most by a factor of,

$$S = \frac{(k) * t_p}{(k - p + 1) * t_p} = \frac{k}{k - p + 1} \qquad p>=2;\ k>=p \qquad (5\text{-}11)$$

Provide Flexibility and Reliability

Flexibility and Reliability are the two aspects should be considered while choosing architecture of development a system especially because the nature of the system will used rapidly, distributed and complex. Choosing the MAS approach for development the proposed IRS provides the flexibility and reliability in operation of the system. In the traditional approach, all tasks are defined as functions or methods, which only permit sequential execution as it shows in Figure 5-1. In such a system, any failure in any part of the system affects the whole system and increasing of data input in one process influences increasing in the following process. In a multi-agent system tasks are operated independently, and can also be operated in parallel. Hence, some system functionality can be maintained if one or more of the task agents failed. For example, if the user modelling (machine learning) task agent failed, the filtering task can still be performed but it will be based on the previous user profile. Furthermore, the flexibility of a multi-agent system approach means that each task agent can be facilitated with a backup agent. Task agents are registered with a facilitator agent responsible for managing all agents' life-cycles. Thus, addition and removal of agents can be easily facilitated. In reality, only compulsory agents are assigned with backup.

Easy of Maintenance

Based on the discussion above shows that the flexibility and reliability criteria cause the easy to maintain the system when there are changes that need to be done in future. The traditional method is maintained by the module or function, which the testing process needs to perform by integrating all the modules. The nature of multi-agents

system comprises of individual objects, each carrying out a specific function. This enables system developers to focus on information-centred, rather than a single, monolithic system. Each function can be implemented and executed independently, so that the testing and maintenance process can performed on the failed agent which does not affect operational of whole integration of agents. This has advantages in development of large and complex system such as the IRS. The MAS is suitable for large system, which each designer focused development is specific agents while other may focused on the designing the interactions of the agents to achieve the business goal.

5.4. Summary

The key issues discussed in this Chapter are the justification for using a multi-agent system approach for development of IRS. From the justification, a methodology for the development process is discussed. Several software development methodologies are reviewed, from which the agent system development process is adopted. The development process consists of requirements analysis, agent system design and agent level design stages. Each stage is provided with the appropriate modelling tools: Use Cases for requirements analysis, and Sequence Diagrams in agent system design and CRC cards in agent level design. The integration of learning strategies for adaptation of the information retrieval system is also discussed.

Chapter 6. System Development and Implementation

6.1. Introduction

The previous Chapter discussed the agent system design phases. This Chapter focuses on the agent environment and agent system deployment (implementation) phases. The purpose of the agent environment phase is to develop an environment that allows agents to be executed. Development of the agent environment is a complex task, because it comprises a combination of several technologies, such as real-time functionality and distribution [Bradshaw 1997]. There are several issues pertinent to multi-agent systems to be considered in order to develop an agent environment including, agent management, security, and communication.

6.2. Agent Environment Development

There are two methods to develop an agent environment. The first method is to develop a multi-agent system (MAS) from scratch. This method is advantageous in terms of accuracy of the environment requirements, but the disadvantages are the development is very complex, time consuming and needs expertise from a variety of areas. There are a few agent-based systems which have been developed in this way, such as CIAgent [Bigus and Bigus 1998], and CASMIRE [Berney and Ferneley 1999]. Most of them only provide functionality limited to their development, rather than reusable objects for development of other multi-agent system. For example, CIAgent provides autonomy but do not provide real distribution. The second method is to use an existing agent environment that has been developed either for commercial or academic purposes. Currently, there are more than 50 different agent environments aimed specific areas of agent application [Tools 2001]. Figure 6-1 shows the process for choosing an agent environment [Eiter and Mascardi 2002, Othman 2003].

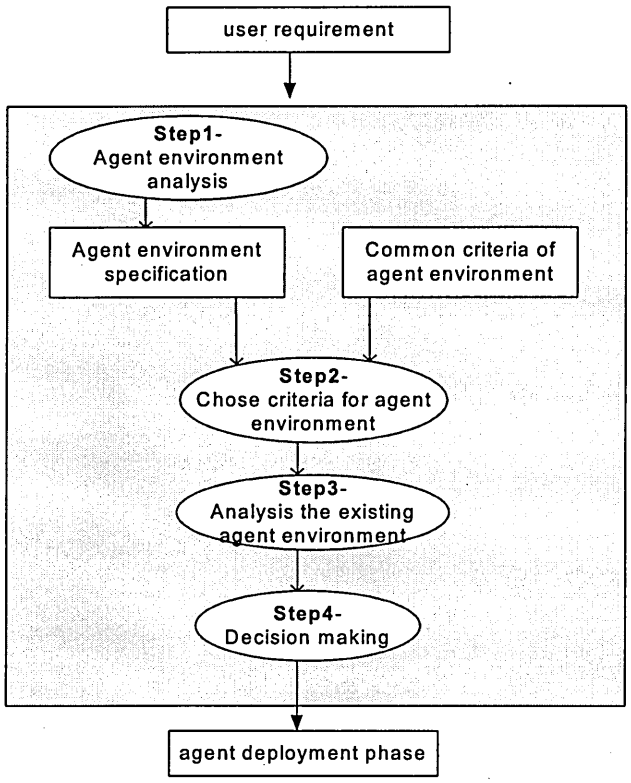


Figure 6-1 : The decision making process of choosing an agent environment

There are several issues that should be considered in agent environment analysis. Gervais [2002] proposed five types of requirements that should be analysed for an agent environment: technology, information, engineering, computation and enterprise. Only three of these have been considered in this research: technology, enterprise and computation. Technology describes the appropriate approach used in the development of agent environment. This includes the distribution and openness, which mainly leads to identification of the appropriate agent architecture. Enterprise is associated with the requirements of a specific deployment such as budget, time and other organisational constraints, while computation refers to the specific techniques for providing agent characteristics.

In selecting criteria for choosing an agent environment Ricordel [2000] provides several guidelines. These are: completeness, applicability, complexity and reusability. Analysis of agent environments is a process of prioritising the criteria for choosing the agent environment.

Analysis of the existing agent environment is to analyse the agent environment that most fulfils the prioritised criteria. At this stage, a comparison should be made between at least three agent environments. Finally, a decision is made on the most appropriate agent environment by selecting the best rated environment, based on ranked criteria. Five agent development environments were evaluated and compared on a number of criteria that included: availability (at commencement of this research study), licensing terms, development language, agent communication language, and applicability. The basic features of the agent environments are briefly presented in the Appendix.

6.2.1. Evaluation of the Agent Environments

In order to choose an appropriate agent environment, the five agent environments were analysed based on several criterion selected from [Ricorrdel 2000, Eiter and Mascardi 2002 and Othman 2003] and prioritised as discussed below:

- The programming languages used for developing the system, such as Java or prolog. Java language provides more functionality to implement agent then Prolog language.
- Availability of the system, whether it needs a license (and terms of license) to open source. This criteria is related to issues of enterprise requirements.
- Support for multi-agent system or single agent, and mobility
- Provision of functionality to easily interact with non-agent software, especially legacy systems.
- Support for ACL.
- Support for standard communication channels of distribution, such as, TCP/IP,
- Provision of functionality for reasoning and type of reasoning offered.
- Type of agent architecture, which represents the paradigm of how agents are designed, such as, BDI and reactive agents.

The five agent environments are ranked according to the above criteria, as shown in Table 6-1. Merit points are awarded on a scale of 1 to 4, where 1 is the worst (least desirable) and 4 is best (most desirable).

Product name	Zeus	Agent Builder	JAFMAS	JATLITE	JADE
Development Language – Java	1	1	1	1	1
Availability / licensing/ Level of open source	3	1	2	2	3
Mobile/Stationary	Stationary(1)	Stationary/Mobile (2)	Stationary(1)	Stationary(1)	Stationary/Mobile(2)
Openness (Support agent in another tools)	2	2	1	1	3
interaction within a non-agent software	2	2	2	2	3
Availability of source/ Reusability	1	1	3	3	3
Agent communication language	KQML/FIPA-ACL(3)	KQML(1)	KQML(1)	KQML(1)	FIPA-ACL(2)
Support distribution (TCP/IP)	3	3	1	4	3
Providing reasoning	2	2	2	1	3
Agent Architecture	BDI	Agent-0/BDI	BDI	Reactive	Reactive
Documentation and Support	2	2	1	1	3
Ease of installation	2	1	1	1	3
Complexity/Ease of implementing agent	3	1	1	1	2
Applicability	2	2	1	1	2
Total Merits	27	21	17	20	33

Table 6-1: Agent Development Environment Comparison

The Table 6-1 shows that Jade has the highest total merits compared to the others. The most significant features were availability, documentation, support during deployment, ease of installation, and freedom to add additional features. On this counts, Jade is the most appropriate for development the IRS system. However there are more reasons for choosing Jade including:

- **Distribution**: because of the complexity of the system it is useful to be able to have distributed agents.
- **Interaction with non-agent software**: it is essential that the agents be able to interact with non-agent software, in particular search engines.
- **Providing reasoning:** in order to have proactive behaviour, it is essential to have reasoning capabilities.

6.2.2. JADE Utilities.

JADE is a framework that simplifies the development of multi-agent systems while ensuring standard compliance with the FIPA specifications. Jade has been developed by a group at CSELT and the University of Parma. The following are the main features of Jade:

- Distributed agent platform. The agent platform can be distributed across machines and the configuration can be controlled via a remote GUI. Agents are implemented as Java threads and live within *Agent Containers* that provide the runtime support to the agent execution.
- FIPA-compliant Agent Platform, which includes the *AMS* (*Agent Management System*), the *DF* (*Directory Facilitator*), and the *ACC* (*Agent Communication Channel*). All these three components are automatically launched at the agent platform start-up and there is no need to explicitly implement them.
- JADE has some graphical tools to support development, as discussed below. These graphic tools are used to simulate the execution of the design agent system, before the deployment phase.

Remote Monitoring Agent (RMA),

- to browse the white-page service and
- to control the agent life-cycle (e.g. remote creation, agent migration)
- to activate/deactivate message transport protocols (MTPs) on containers
- to browse white-page services of remote agent platforms

Directory Facilitator Graphic User Interface(GUI)

- to browse the yellow-page service
- to make DF federations and browse remote DF's

Dummy Agent

- send/receive store/save ACLMessages

Sniffer Agent

- to sniff, debug, save to file, multi-agent conversations

Introspector Agent

- to debug an agent: queue of sent/received messages, queue of behaviours,.

Other extra functionality that Jade offers are:

- Intra-platform agent mobility, including state and code of the agent.
- Support to the execution of multiple, parallel and concurrent agent activities via the behaviour model. JADE schedules the agent behaviours in a non-preemptive fashion.
- Many FIPA-compliant DFs can be started at run time in order to implement multi-domain applications, where a domain is a logical set of agents whose services are advertised through a common facilitator. Each DF inherits a GUI and all the standard capabilities defined by FIPA (i.e. capability of registering, deregistering,

modifying and searching for agent descriptions; and capability of federating within a network of DF's).

- Efficient transport of ACL messages inside the same agent platform. In fact, messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures. When crossing platform boundaries, the message is automatically converted to/from the FIPA compliant syntax, encoding, and transport protocol. This conversion is transparent to the agent implementers, which need to deal only with Java objects.
- Library of FIPA interaction protocols ready to be used.
- Automatic registration and deregistration of agents with the AMS.
- FIPA-compliant naming service: at start-up agents obtain their GUID (Globally Unique Identifier) from the platform.
- Support for application-defined content languages and ontologies.
- In-Process Interface to allow external applications to launch autonomous agents.
- *JessBehaviour* that allows full integration with JESS. JESS is a scripting environment for rule programming written in Java offering an engine using the Rete algorithm to process rules. Therefore, while JADE provides the shell of the agent and guarantees the FIPA compliance, JESS allows the use of rule-oriented programming to define agent behaviours and use its engine to execute them.

6.2.3. Agent System Deployment using JADE

The concern in this section is with reuse of the Jade components, and how to incorporate then into the design of the IRS. The deployment process adopted object-oriented concepts. Jade has several reusable components such as 'Jade.core', 'Jade.onto', 'Jade.domain', 'Jade.gui', 'Jade.proto' and Jade wrapper. These components cover all the above features for developing agent-based systems, discussed above.

1. Creating agent.

'Jade.Core'- This creates an agent, which creates a class-name and inherits the agent class.

2. Creating behaviour

   'Jade.core.behavior' – This is a sub-component for the agent. It is used for creating roles or tasks for each agent. The task generates action for communication.

   Agent-specific tasks are implemented by writing one or more *behaviour* (from 'Jade.core.behaviour') subclasses, instantiating them and adding the behaviour objects to the agent. User defined agents inherit from the *Agent* class the basic capability of registering and deregistering with their platform and a basic set of methods that can be called to implement the custom behaviour of the agent (e.g. send and receive ACL messages, use standard interaction protocols, register with several domains).

   JADE contains ready-made behaviours through a set of abstract classes for the most common tasks in agent programming, such as sending and receiving messages. Jade offers the developer different class behaviours such as SimpleBehaviour, OneShotBehaviour, CyclicBehaviour, CompositeBehaviour, and SequentialBehaviour [Bellifemine and Poggi 2001].

3. Creating communication

   The 'Jade.lang' component for communication consists of two sub components: message using ACL, and ontology ('Jade.onto' is a component for defining different ontology between two agents). Jade also provides a ready-made ACL message class for agent communications. The class contains a set of attributes as defined by FIPA specification. An agent wishing to send a message should create a new ACLMessage object, fill its attributes with appropriate values, and finally call the method Agent.send(). Likewise, an agent willing to receive a message should call receive() or blockingReceive() methods, both implemented by the Agent class .

   Furthermore, this class also defines a set of constants that should be used to refer to the FIPA performatives, i.e. REQUEST, INFORM. When creating a new ACLMessage object, one of these constants must be passed to the ACLMessage class constructor, in order to select the message performative.

ACL message class contains a Message Template class to build patterns to match ACL messages against. Using the methods of this class the programmer can create one pattern for each attribute of the ACLMessage.

Other components are 'Jade.gui', components of tools or utilities that are used to test prototypes of agent communication, but are not part of the reusable components for deployment. It is similar to Jade wrapper.

The next section shows the implementation of each agent in the deployment, concepts described above were integrated into the individual agent design.

6.3. Implementation of the Proposed System

In this section, more detail is given of each individual agent proposed in the previous Chapter. All agent classes are in one package, called Agents. The package consists of five individual agents, namely, user agent, search agent, user-model agent, document agent and filter agent (Figure 6-2). The following subsections describe the details of each agent.

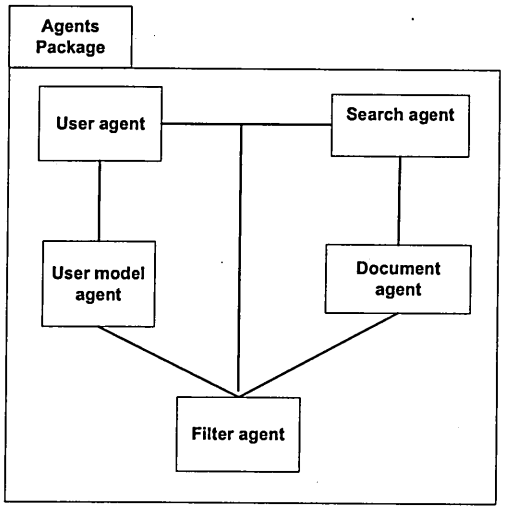


Figure 6-2: Agents Package Diagram

6.3.1. User Interface Agent

This agent is responsible for managing the dialog between the user and the system. It allows the user to enter keyword based queries. The Use Interface agent communicates

with the Search agent, User model agent and Filter agent. User interface agent includes UserLoginBehaviour and one sub behaviour called ShowingResultBehaviour. The following piece of code demonstrates how an agent and its behaviour can be created.

```
public class UserAgent extends Agent {
/** UserAgent **/
public UserAgent() { }
public void setup(){
    addBehaviour(new UserLoginBehaviour(this));
  } //setup
} //useragent class
```

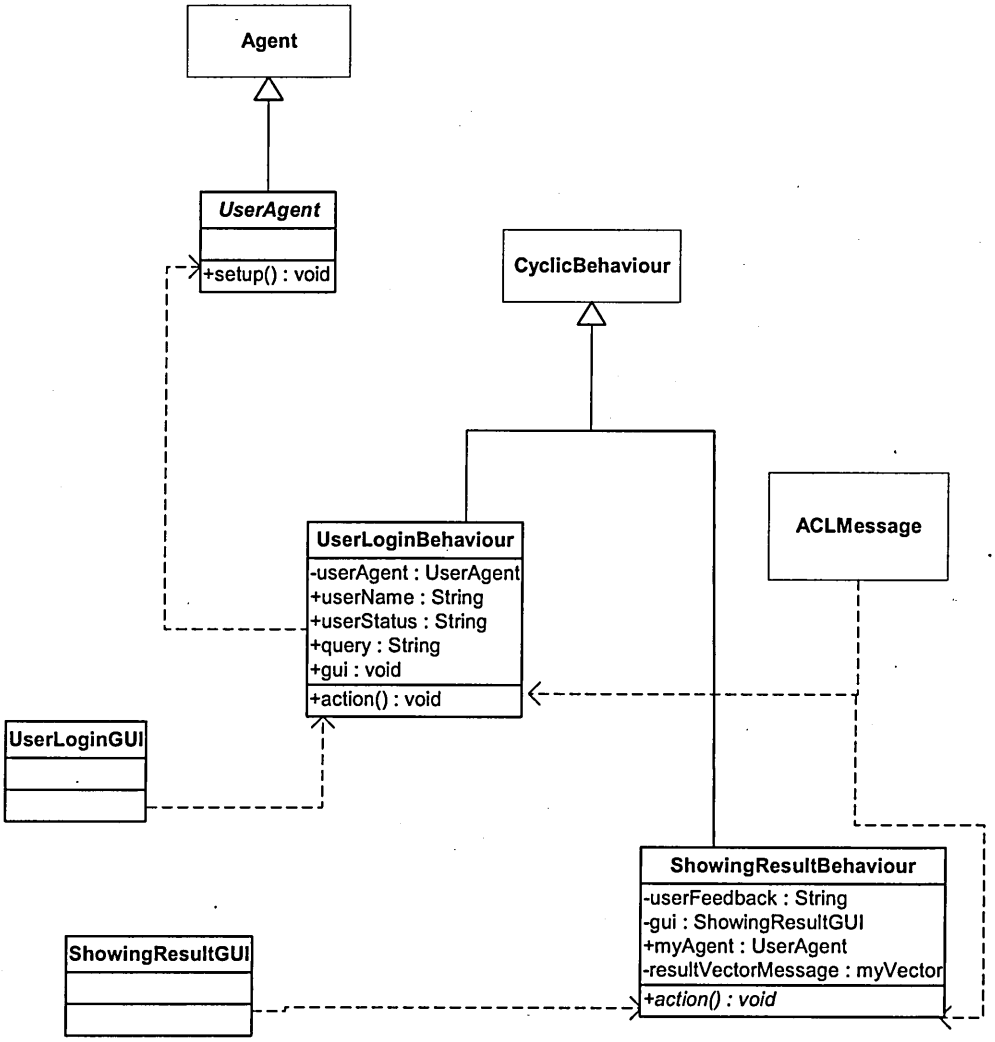


Figure 6-3: User Interface Agent class diagram

UserLoginBehaviour

The dialog box appears on the screen when the system starts and then the user must enter a user name, user status (new user or old user) and keywords to log in. The Figure 6-4 below is a screen shot of the user login dialog box.

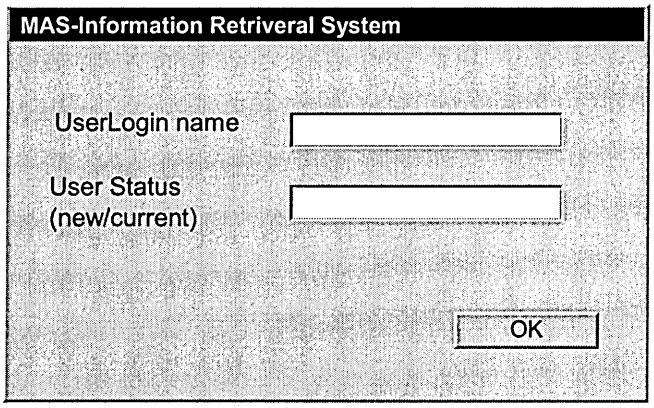


Figure 6-4 : User login dialog box

The above task is a UserLoginBehaviour which is an extension of CyclicBehavior. The following activity diagram (Figure 6-5) shows the userLoginBehaviour process. One of the important parts of any behaviour is managing receiving/sending messages to the other agents. In this behaviour one request message has been created and sent to the user model agent. The following piece of code shows how we can implement the creation and sending of a request message.

```
if ( userstatus.equalsIgnoreCase("new") )  {
    ACLMessage msg = new ACLMessage("request");
    msg.setDest("UsermodelAgent");
    msg.setLanguage("A1CL");
    msg.setContent(createActionContent("UsermodelAgent", "createinit", userInfo));
    userAgent.send(msg);
} //end if new
     if ( userstatus.equalsIgnoreCase("old")) {
     ACLMessage msg = new ACLMessage("request");
```

```
    msg.setDest("UsermodelAgent");
    msg.setContent(createActionContent("UsermodelAgent", "applyGA", userInfo));
    msg.setLanguage("A1CL");
    userAgent.send(msg);
} //end if  old
```

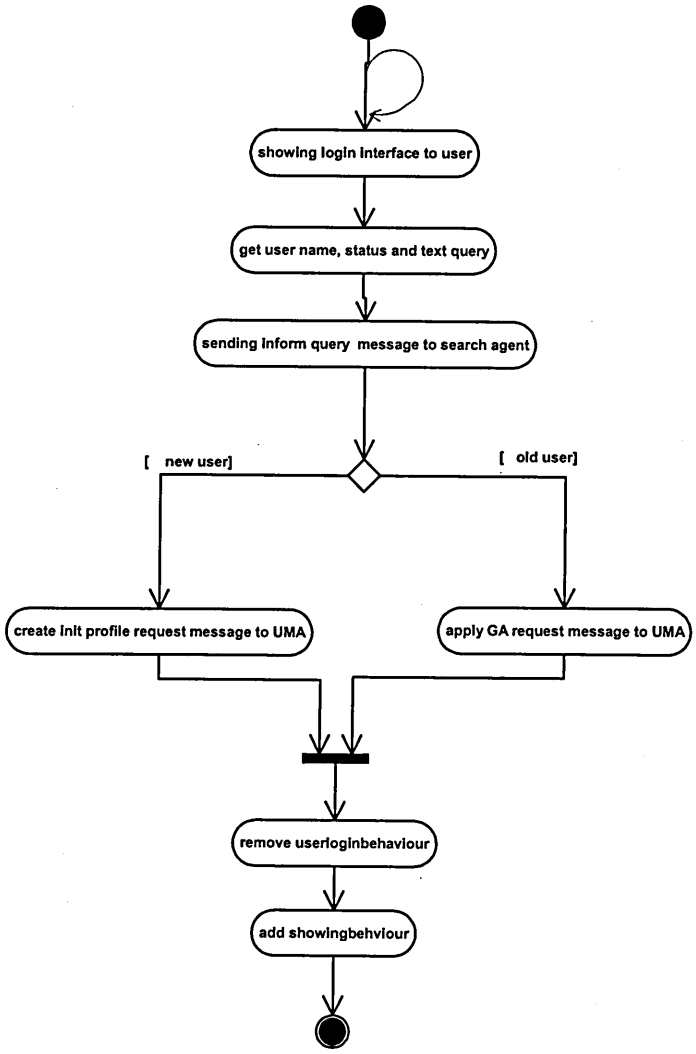


Figure 6-5: UserloginBehaviour activity diagram

ShowingResultBehaviour

When the filter agent has completed the search process, the result is sent to the User Interface agent for display to the user. The system also enables the user to evaluate the results, via presenting options and allowing selection to be made from list of choices. These tasks are included in the ShowingResultBehaviour, which is sub-behaviour of UserLoginBehaviour. ShowingResultBehaviour is also an extension of CyclicBehaviour. Figure 6-6 below is a screen shot of result dialog box.

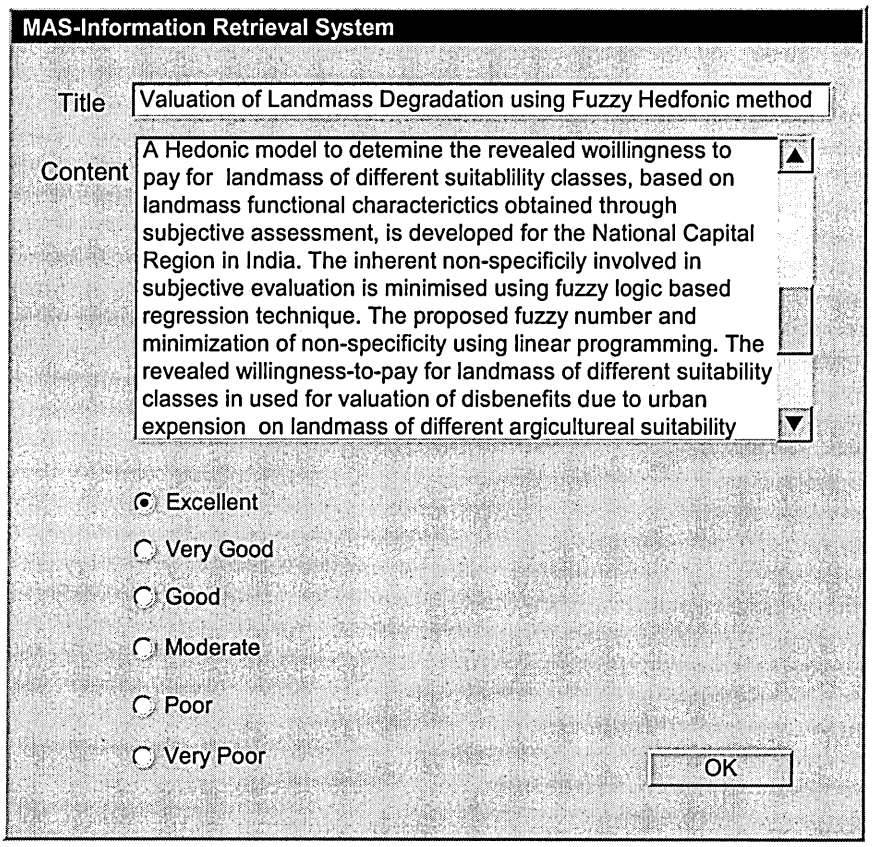


Figure 6-6 : Display results dialog box

Figure (6-7) shows the activity diagram for the ShowingResultBehaviour process.

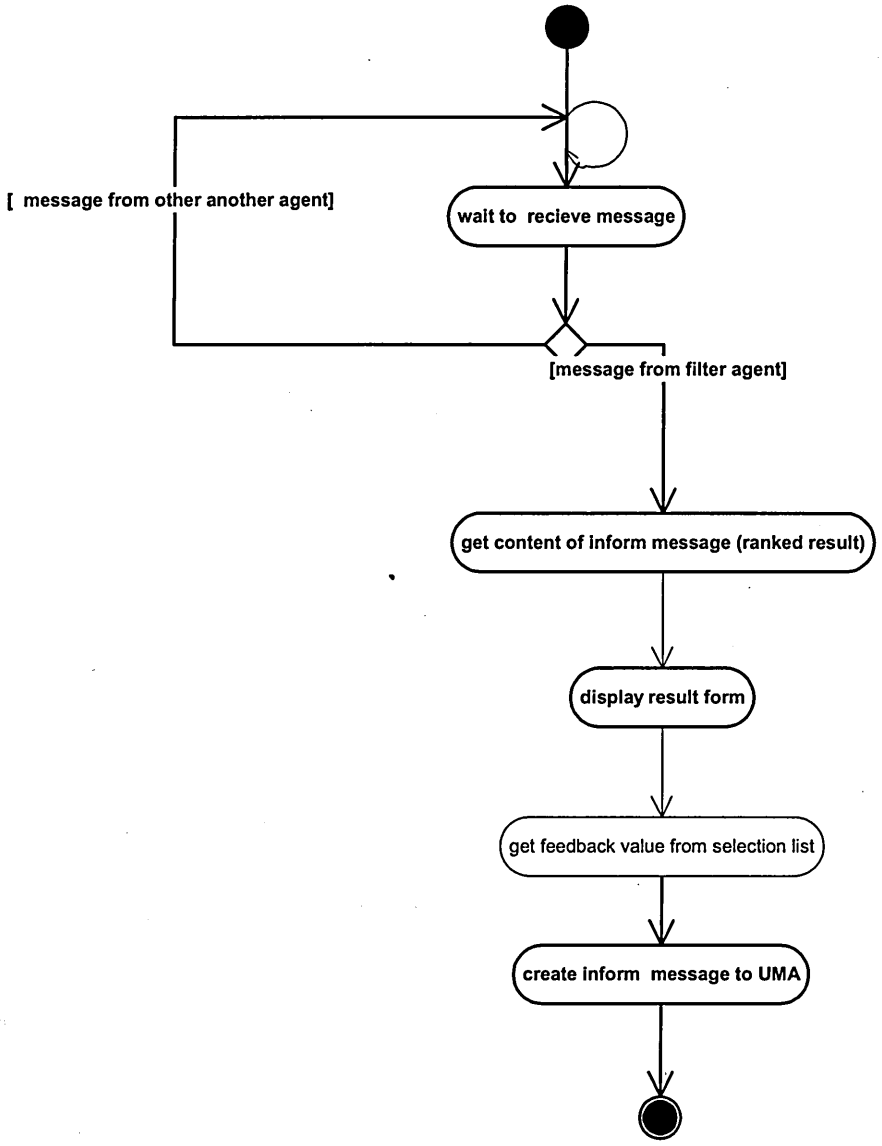


Figure 6-7: ShowingresultBehaviour activity diagram

6.3.2. Search Agent

The search agent performs the task of submitting queries in the correct manner and gathering information from the source (e.g. Internet). The results of this task are then transferred to the document agent in the form of the inform message type. The Search agent consists of one cyclic behaviour, named SearchBehaviour. Figure 6-8 shows the

Search Agent class diagram. Figure 6-9 shows the activity diagram of SearchBehaviour process.

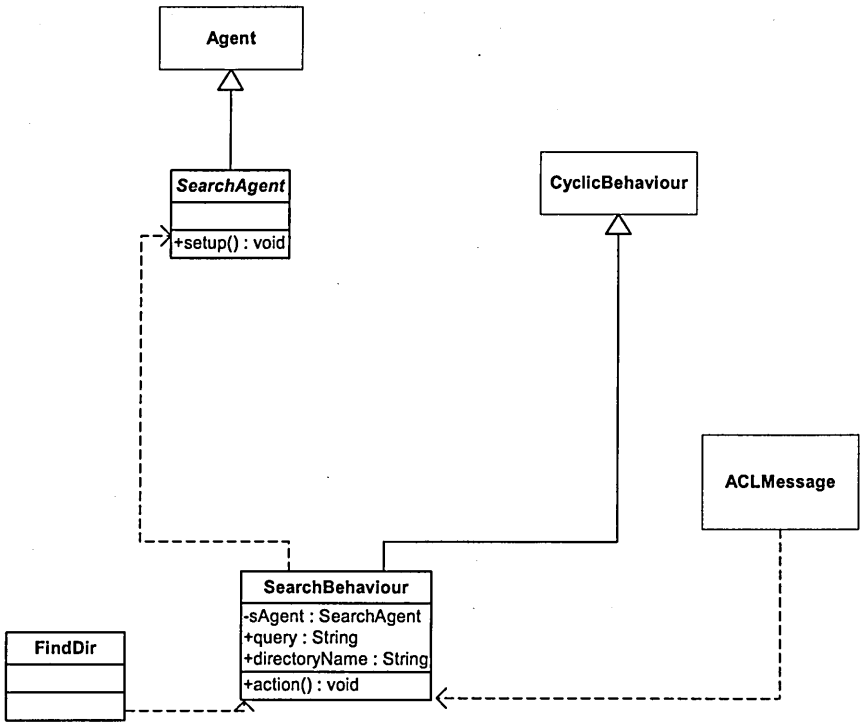


Figure 6-8: Search Agent class diagram

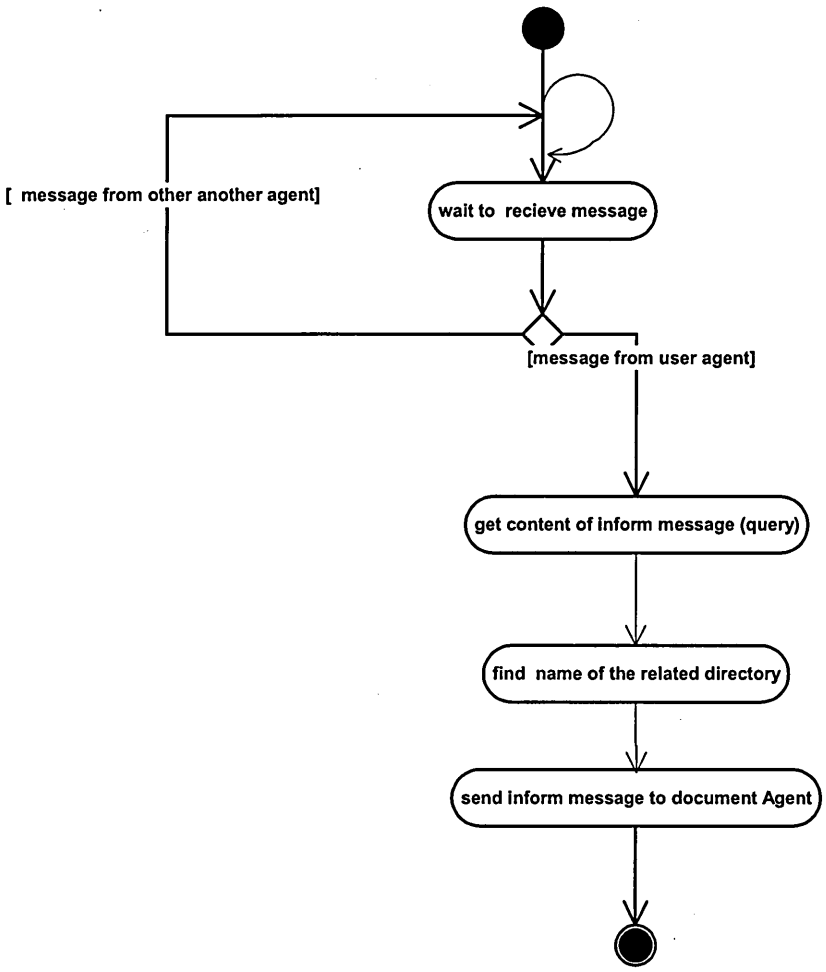


Figure 6-9: SearchBehaviour activity diagram

The following piece of code shows how inform-message is implemented and its attributes in this behaviour set.

```
ACLMessage msg5 = new ACLMessage(ACLMessage.INFORM);
 msg5.addDest("DocumentAgent");
 msg5.setContent(directoryname);
 SAgent.send(msg5);
```

6.3.3. Document Agent

The goal of this agent is to create a representation of each document or, in other words, to index the documents. The representation typically used is a set of features derived from the document collection, which is a vector of weighted keywords. The Search agent consists of one cyclic behaviour, named IndexBehaviour. Figure 6-10 shows the Document Agent class diagram. The result of Indexing behaviour is sent to Filter Agent via inform-message.

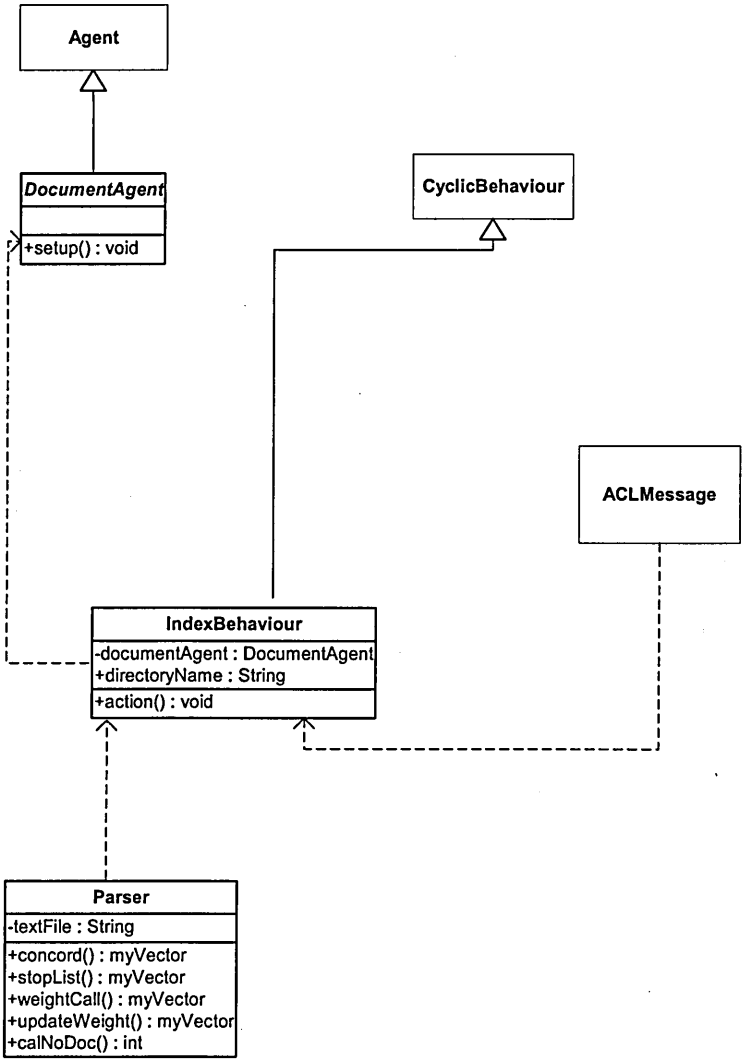


Figure 6-10: Document Agent class diagram

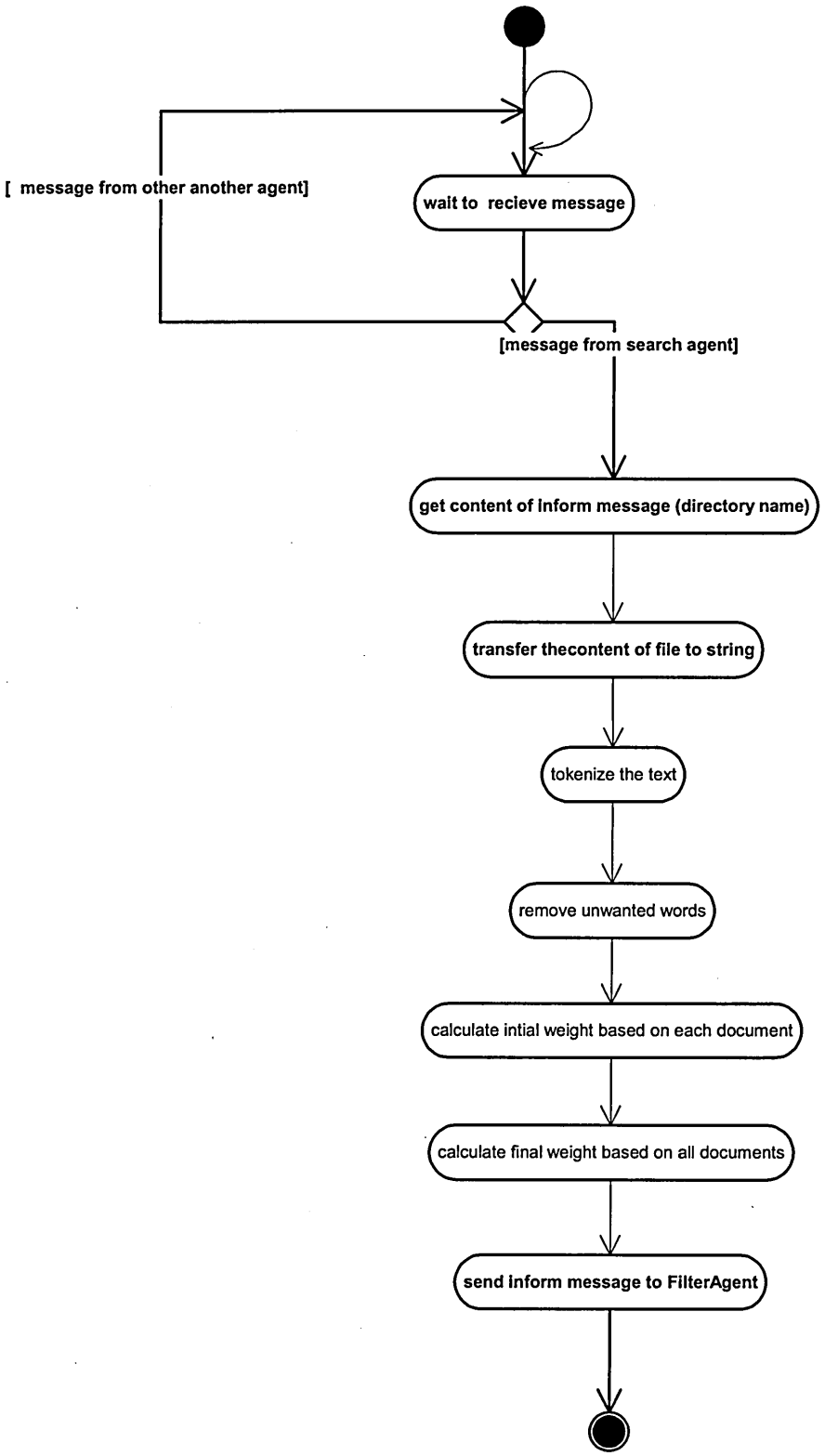


Figure 6-11: IndexBehaviour activity diagram

6.3.4. User-model Agent

The remit of this agent is to guide the user in the query formulation process and to store and manage the user's interests in the form of a user profile. This agent is a key element of the system architecture and is composed of genetic algorithms (GA), relevance feedback (RF) and a fuzzy logic component.

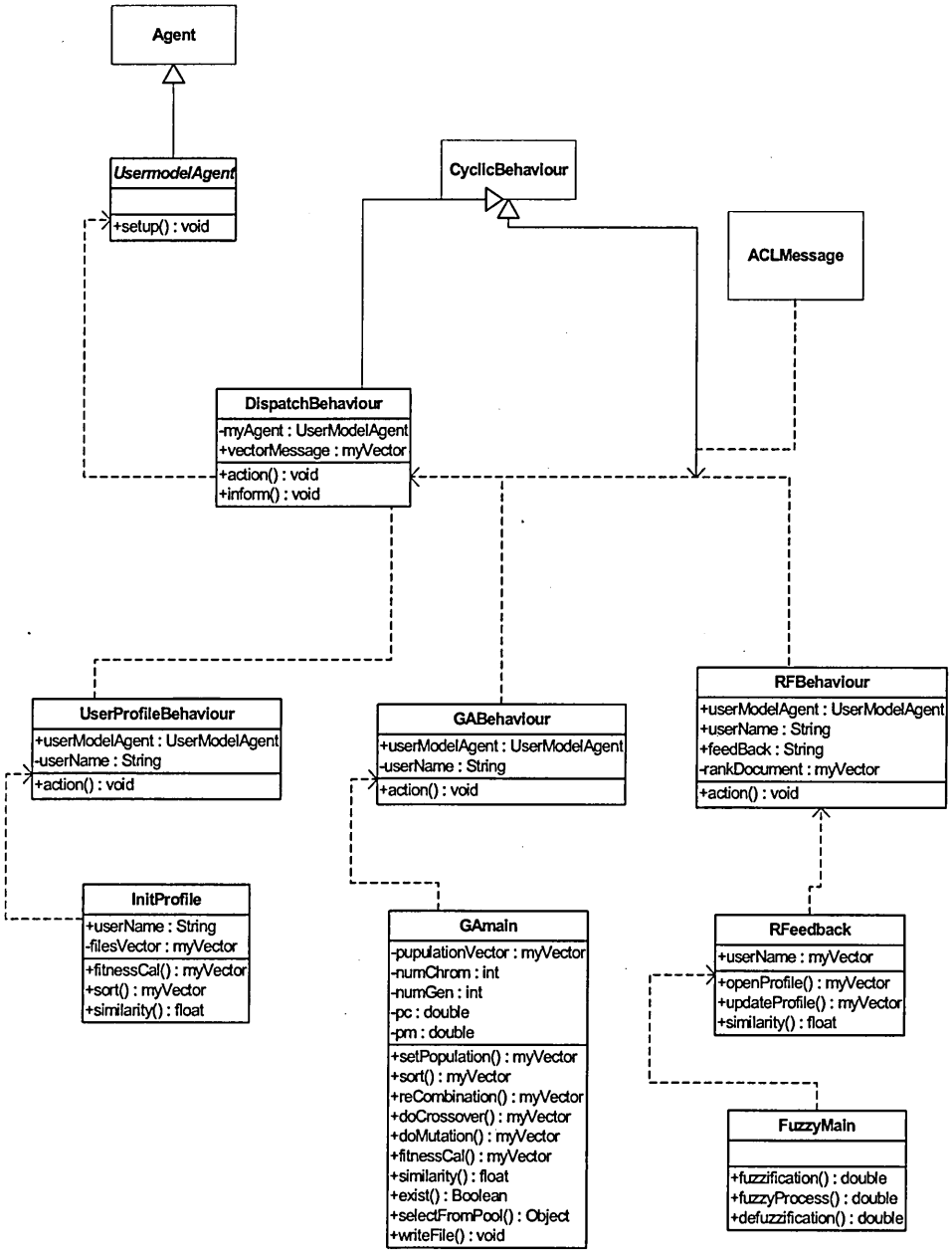


Figure 6-12: User model Agent class diagram

The User-model Agent includes several behaviours such as DispatchBehaviour, UserProfileBehaviour, GABehaviour and RFBehaviour. Figure above (6-12) shows the User model Agent class diagram.

DispatchBehaviour

Due to the variety of messages which come to the user model agent, there is a need to distinguish the message and act accordingly using a template. The following piece of code shows the template message creation (see Figure 6-13).

```
ACLMessage msg = myAgent.receive();

    MessageTemplate  mt1 = MessageTemplate.MatchSource("FilterAgent1");
    MessageTemplate  mt2 = MessageTemplate.MatchSource("UserAgent");

if((mt2.match(msg))) {

      if(msg.getLanguage().equals("A1CL1")){
            vector_message = (Vector) msg.getContentObject();
            Rank_document= (Vector) vector_message.elementAt(0);
            userfeedback = (String) vector_message.elementAt(1);

            myAgent.addBehaviour(new RFBehaviour(myAgent, username, userfeedback,
Rank_document));

      } //end if lang  alcl1

if(msg.getLanguage().equals("A1CL")){
          CLParser parser = CLParser.create();
```

```
Action action = (Action) parser.parse(new StringReader(msg.getContent()),
msg.getType());
String actor = action.getActor();
String action_type = action.getActionType();
int numberOfParameters = action.getNumberOfActionParameters();
Concept parameter = (Concept) action.getActionParameter(0);
username = parameter.getAttributeValue("username").toString();

//********************** new user*******************************

if(action_type.equals("createinit")){

myAgent.addBehaviour(new UserprofileBehaviour(myAgent, username));

  } //if actiontype    */

//************************old user ***************************

 if(action_type.equals("applyGA")){
  myAgent.addBehaviour(new GABehaviour(myAgent, username));

        } //if actiontype    */

    } //end if getlanguge
```

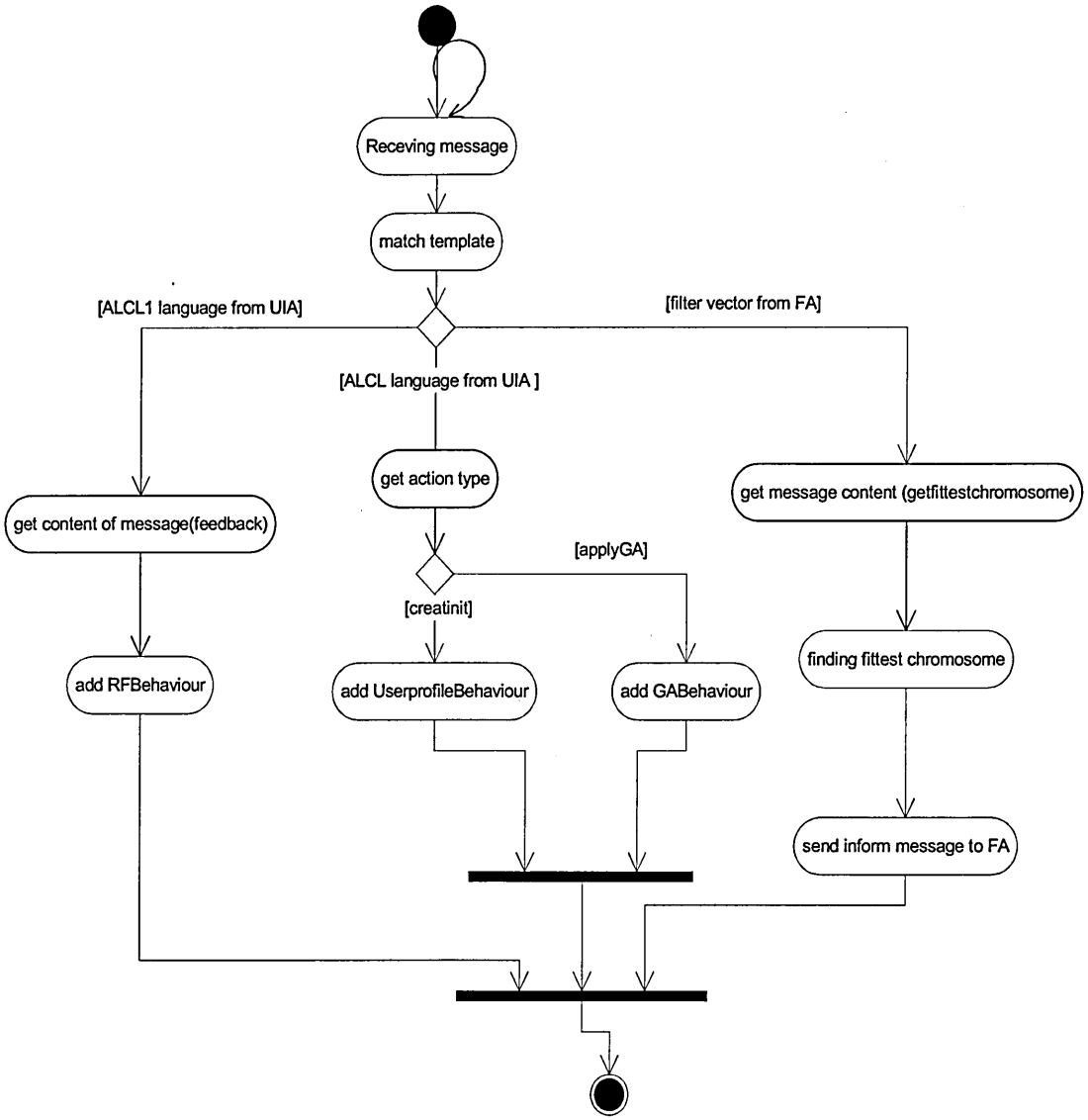


Figure 6-13: DispatchBehaviour activity diagram

UserProfileBehaviour

This behaviour is responsible for creating user profile for the new user by selecting documents randomly (Figure 6-14).

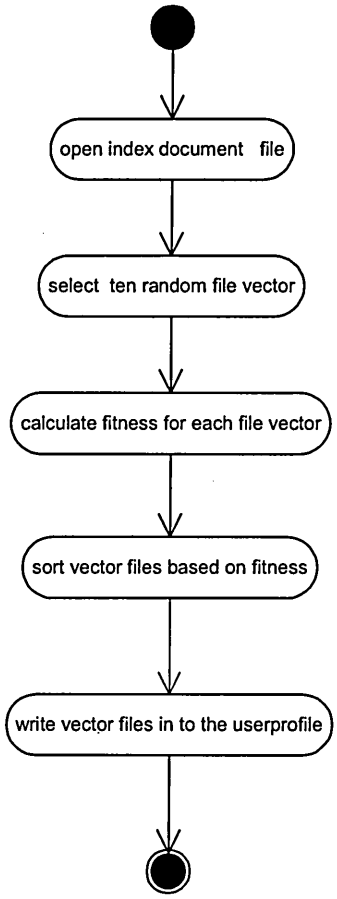


Figure 6-14: UserprofileBehaviour activity diagram

GABehaviour

This behaviour is responsible for applying all aspects of GA, such as, crossover, mutation and recombination to existing users profile (Figure 6-15).

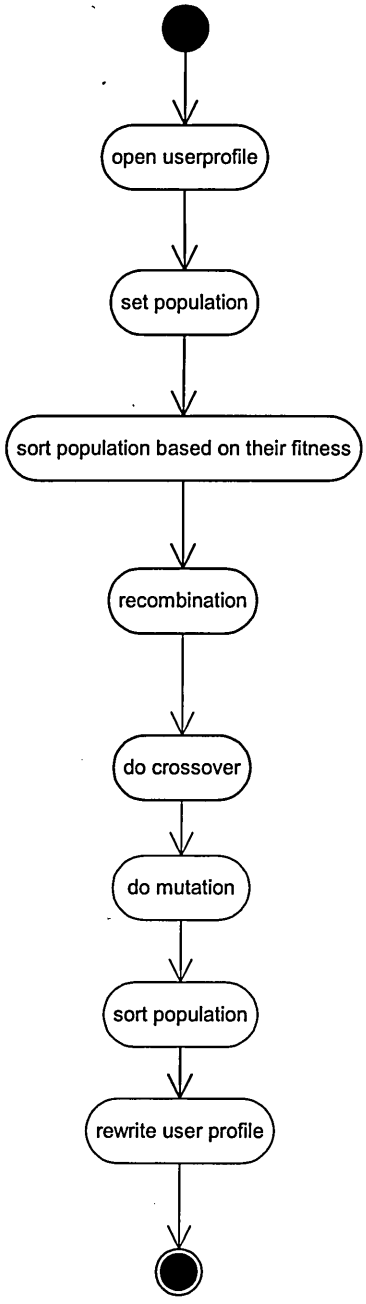


Figure 6-15: GABehaviour activity diagram

RFBehaviour

The RFBehaviour updates user profiles based on feedback received from the user (Figure 6-16). To achieve this task, fuzzy logic has been applied to calculate the relevant parameters such as alpha value, described in Chapter 5.

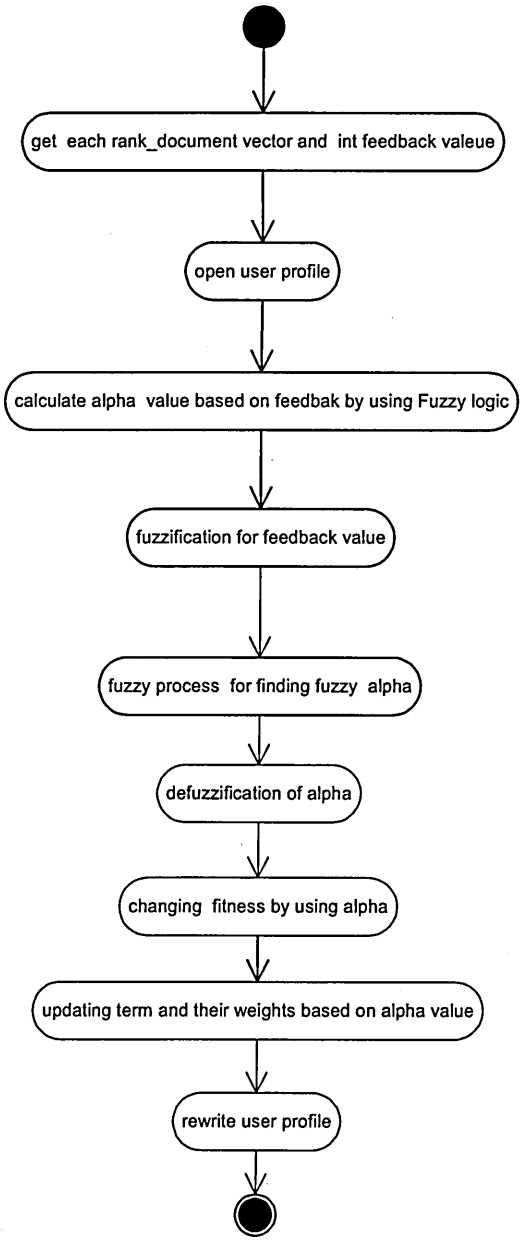


Figure 6-16: RFBehaviour activity diagram

6.3.5. Filter Agent

The Filter agent (Figure 6-17) increases the retrieval precision by ranking documents according to user preferences. In order to do this task, the system needs to assign a similarity measure to each document that indicates how closely it matches a query. This process can be formalized as an inner product of a query vector (selected chromosome in user model) with a set of documents vector (for details see Chapter 5).

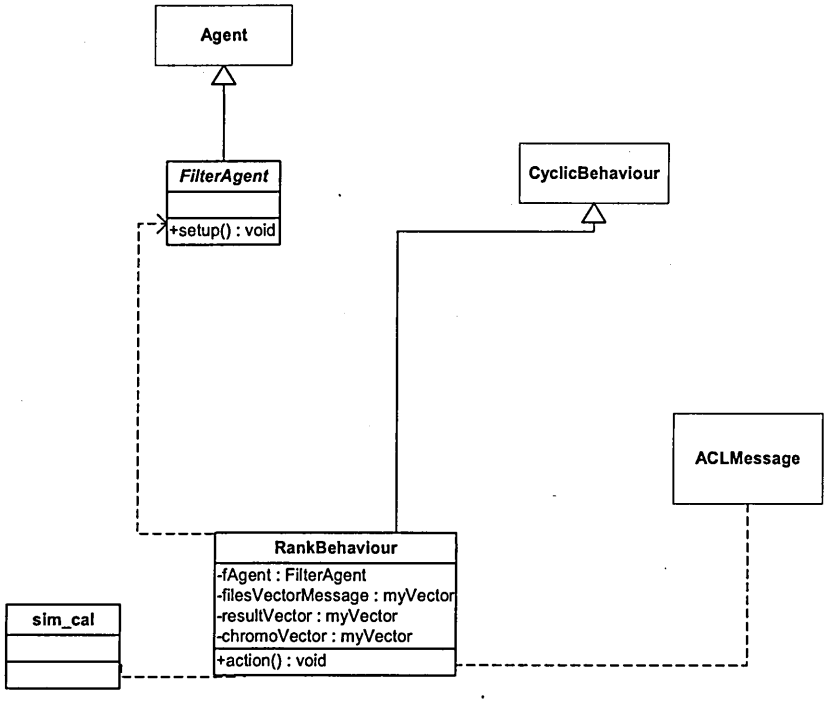


Figure 6-17: FilterAgent class diagram

RankBehaviour

When RankBehaviour receives an inform message from Document Agent, which includes the list of index of documents, the agent sends a request message to User Model Agent. The request message asks for the fittest chromosome in the user profile to be able to rank the index documents. The activity diagram for RankBehaviour is shown in Figure 6-18.

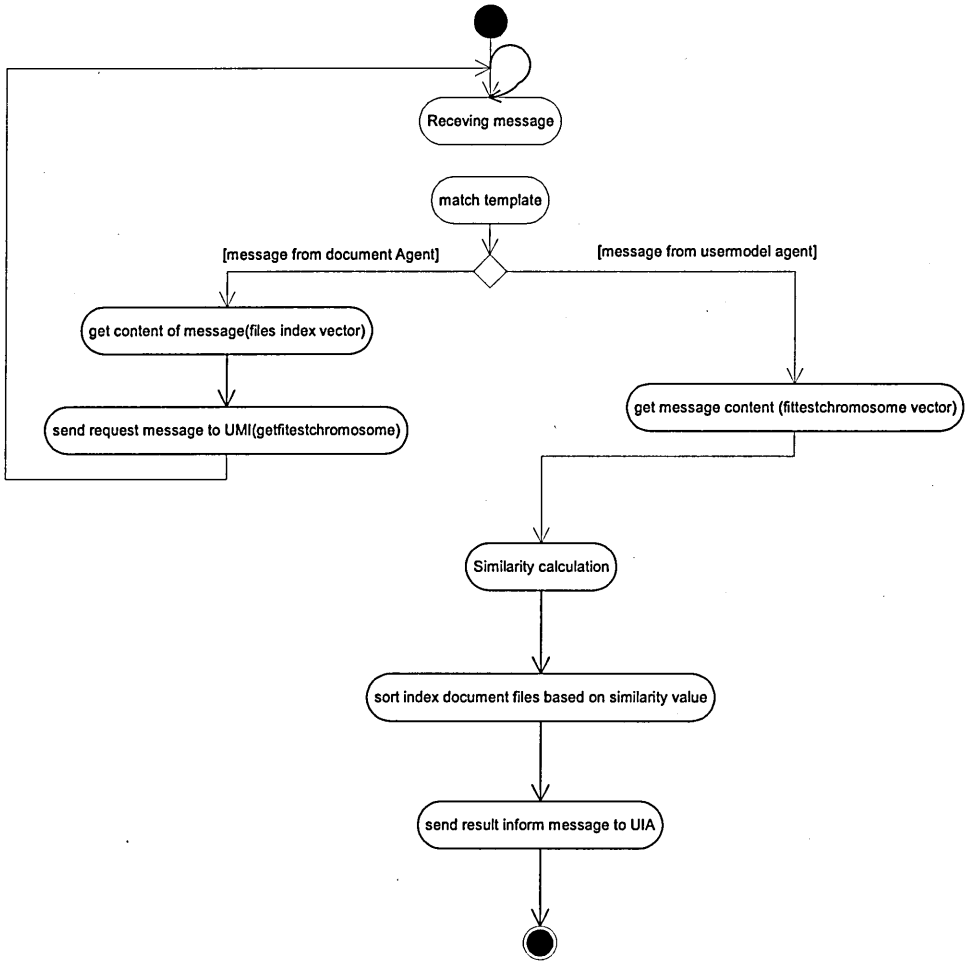


Figure 6-18: RankBehaviour activity diagram

6.4. Summary

This Chapter has presented an overview of the actual implementation of the proposed information retrieval system. Firstly, selection of an agent system development environment was investigated. An analysis and comparison of five common agent development platforms was carried out, from which justification was shown for the selected environment. A multi-agent system was specified to satisfy the functionalities of the proposed IRS. Characteristics, behaviour and deployment of each of the agents in the IRS was discussed, using class and activity diagrams.

Chapter 7. SYSTEM EVALUATION

7.1. Information Retrieval Evaluation

With the continued information explosion, including the emergence of the Internet and digital library initiatives, information retrieval (IR) evaluation has increased in importance and is an active area of research and development. IR evaluation aims to find appropriate measurement techniques to be used to determine if one approach is better than another. IR research has primarily consisted of two paradigms: systems-oriented research and user-oriented studies. The systems-oriented approach includes IR algorithm development and evaluation and, to some degree, human-system interaction [Reid 2000]. User studies include human behaviour and information seeking research. These paradigms have resulted in new IR algorithms and insights into IR processes. However, problematic issues concerning IR evaluation exist in both communities [Wu and Sonnenwald 1999]. There are different criteria and measurement methods based on IR dependency issues: evaluation of indexing method, document analysis, use of thesaurus, databases, GUI design. A good overview on IR evaluation and its impacts is given by Saracevic [1995] and Sparck_Jones and Willett [1997]. Sparck-Jones [1981] expresses the difficulties with IR evaluation in the following statement:

> *"The problem is really that information retrieval systems are so complicated, and so little understood, and there is such a lack of solid theory about them, that really high class experiments can hardly be expected."*

Information retrieval evaluation dates back to 1950s with the Carnfield test. In the test Cleverdon listed six main measurable quantities [Van Rijsbergen 1979]:

(1) The *coverage* of the collection, that is, the extent to which the system includes relevant matter;
(2) the *time lag*, that is, the average interval between the time the search request is made and the time an answer is given;
(3) the form of *presentation* of the output;
(4) the *effort* involved on the part of the user in obtaining answers to search requests;

(5) the *recall* of the system, that is, the proportion of relevant material actually retrieved in answer to a search request;

(6) The *precision* of the system, that is, the proportion of retrieved material that is actually relevant.

Saracevic [1995] also proposes six levels of IR evaluation: engineering, input, processing, output, use and user, and social. On the engineering level, questions concerning hardware and software performance are addressed. Thus, computational effectiveness and efficiency of given retrieval methods and algorithms are investigated. On the input level, questions about the inputs to and contents of the system are investigated. Thus, questions about coverage in the designated area are asked. On the processing level, questions about the way the inputs are processed arise. Thus, the performance of algorithms, techniques, approaches, and the like are assessed. On the output level, questions about interactions with the system and obtained output(s) are addressed. Thus, the evaluation criteria include assessment of searching, interactions, feedback, and given outputs. On the use and user level, questions of application to given problems and tasks are raised. The assessment criteria are market and fitness-of-use. And finally, on the social level, issues of impact on the environment(s) occur because relevance is situational, meaning information needs are unique to a particular person at particular time. The evaluation criteria include effects on research, productivity and decision-making.

The Text Retrieval Conference (TREC) series, which started in 1992, combines many efforts to provide common performance tests. The TREC project provides a focus for these activities and is the worldwide standard in IR. It also brings academic and commercial developers together in a new dynamic for the field.

In TREC, new types of IR tests, called tracks, are proposed each year, and each track is designed to address a different IR problem. Researchers or research groups may select which track(s) they wish to participate in (Figure 7-1).

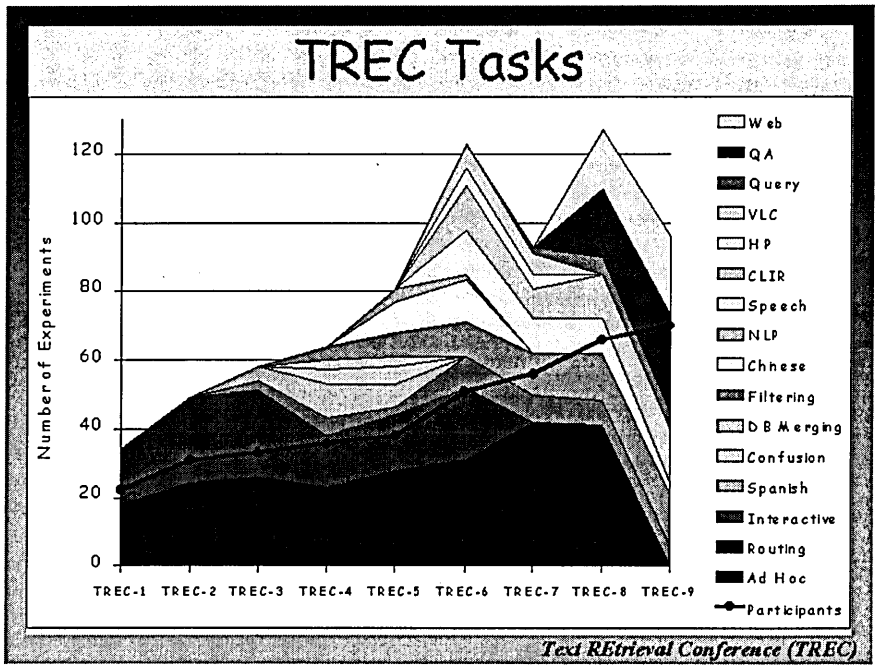


Figure 7-1: Overview on TREC tasks

In TREC 9, the tracks included [TREC 2000]:

(a) *Cross-language track:* The challenge is to retrieve relevant documents regardless of language.

(b) *Filtering track:* The topics are stable and some relevant documents are known. The system must make a binary decision as to whether new, incoming documents are relevant to the topics.

(c) *Interactive track*: This track is used to study user interaction with IR systems, for example, the role of relevance feedback in IR.

(d) *Query track:* This track examines the effects of query variability and analysis on retrieval performance.

(e) *Question answering track:* This is a new track that strives to address "information" retrieval as compared to "document" retrieval. For a set of 200 questions, systems must produce text extracts that answer the questions. The text extracts range from short phrases (2-3 words) to an entire document (1,000 words.)

(f) *Spoken document retrieval track:* This track investigates systems' ability to retrieve spoken documents (recordings of speech.)

(g) *Web track:* In this track, the document set is 2 gigabytes of web data. This track will encourage researchers to investigate whether links can be used to enhance IR.

Contributions from TREC include the development of a community of researchers who compete and compare their algorithms on a regular basis using large document collections. This has led to refined algorithms that produce improved recall and precision measurements, and work well with very large collections of documents. It has also broadened IR evaluation research to include cross-language IR, spoken document retrieval, and the investigation of relevance feedback.

Evaluation of retrieval system is concerned with how well the system satisfies users, not just in individual cases, but collectively, for all users. There are three main ingredients of a meaningful information retrieval experiment: a measure of the effectiveness of the search, a test collection, and a test of statistically significant difference between methods [Hull 1993].

7.2. Measures of Retrieval Effectiveness

The evaluation of an IR system calls upon many research areas, such as Human Computer Interaction (HCI), algorithms and statistics [Reid 2000]. The only aspect of evaluation that concerns this research, however, is the evaluation of the quantity of relevant documents a system retrieves with respect to a query. This aspect of evaluation is known as retrieval effectiveness. The *effectiveness* of an IRS expresses how well the produced output, i.e. the (ranked) list of retrieved items satisfies an information need. This is accomplished by retrieving as many useful items and as few useless ones as possible. The usual measures of retrieval effectiveness of IRS are *recall* and *precision* [Salton and McGill 1983]. Both measures are based on the user's relevance assessments following the retrieval process. *Recall* measures the completeness of the output, that is, the ability of the system to retrieve useful (relevant) information. *Precision* measures the relevance of the output that is the ability of the system to reject useless (irrelevant or noisy) material. A good IRS should exhibit both high recall and high precision. [Van Rijsbergen 1979] proposes another measure, *fallout*, defining it as the ratio of retrieved

non-relevant documents to the total number of non-relevant documents. Jarvelin and Kekalainen [2000] also proposed effectiveness measures called cumulated gain by document rank (CG) and cumulated gain with discount by document rank (DCG). They both measure a combined document rank based on the probability of relevance and degree of relevance.

7.2.1. Recall and Precision

Figure 7-2 illustrates the computation of the *recall* and *precision* measures. For a given query and from the point of view of a user's assessment of relevance, an information item collection C is composed of two disjoint sets: the set R^+ of relevant information items and the set R^- of irrelevant information items. The list r of retrieved information items is composed of the disjoint sets A of relevant information items and B of irrelevant information items.

Recall is the proportion of relevant material retrieved, while *precision* is the proportion of retrieved material that is actually relevant:

$$precision = \frac{|A|}{|r|} = \frac{Number\ of\ relevant\ Information\ items\ retrieved}{Total\ number\ of\ items\ retrived} \quad (7\text{-}1)$$

$$recall = \frac{|A|}{|R^+|} = \frac{Number\ of\ relevant\ Information\ items\ retrieved}{Total\ number\ of\ relevant\ in\ the\ collection} \quad (7\text{-}2)$$

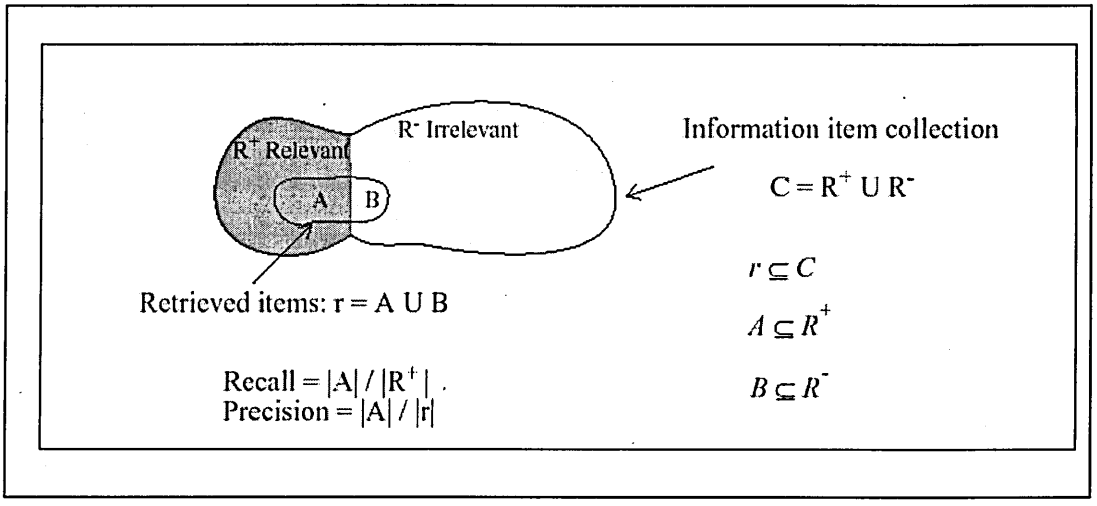


Figure 7-2: Measure of retrieval effectiveness (recall and precision

The precision, $\Pi(\rho)$, is given for different values of recall, ρ. The construction of the recall-precision graph is described in Figure 7-3 [Van Rijsbergen 1979]. One common way to express retrieval effectiveness is the recall-precision graph (Figure 7-4). A typical way to present the result of evaluation in the Precision-Recall curve is discussed in detail by [Keen 1997].

Input: For each query $q \in Q$ the ranked list [$item_1$, $item_2$, $item_3$,.... $item_{kq}$] of retrieved components containing all retrieved items.

Output: The average precision function $\Pi(\rho)$.

Procedure:

1. For each query $q \in Q$ and i= 1, k_q, calculate the pair (π_i, ρ_i). ρ_i is the recall and π_i the precision in the ranked list [$item_1$, $item_2$, $item_3$,.... $item_{kq}$] according the formulas in equation (7-1) and (7-2).
2. For each query $q \in Q$ a function Π_q is defined that assigns to each recall value ρ $\in$ [0,1] the corresponding precision value in the following way:

$$\Pi_q(\rho) = \max [\pi_i \mid (\rho_i \geq \rho)]$$

(The objective is to replace a sawtooth curve by a monotonically decreasing curve where each recall value corresponds to a unique precision value).

3. The average precision function is obtained from:

$$\Pi(\rho) = \frac{1}{|Q|} \sum \Pi_q(\rho)$$

Figure 7-3: Algorithm for the construction of recall-precision graph

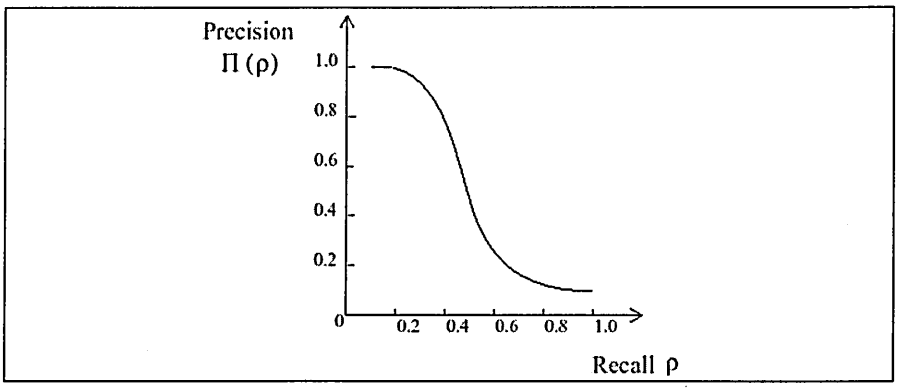


Figure 7-4: Typical average recall-precision graph

In typical keyword-based IRS, *recall* can be increased as the number of retrieved items increases, because some of the new retrieved items can be relevant. At the same time, the precision is likely to decrease, as is shown in Figure 7-4. Usually, mechanisms improving recall affect precision, and vice versa. A trade-off is often necessary between those mechanisms, to control retrieval effectiveness.

Let r_i be the number of relevant documents among the top i documents in the returned list, and r_q be the number of relevant documents to the query in the collection. The precision for the top i documents, π_i, is defined as:

$$\pi_i = \frac{r_i}{i} \tag{7-3}$$

The recall is then defined as:

$$\rho_i = \frac{r_i}{r_q} \tag{7-4}$$

For collection with multiple queries, performance can be evaluated by measuring precision at several different levels of recall. This can be done for each query separately and then averaged over all queries. Another measure of retrieval effectiveness is *average precision.* The N-point interpolated average precision for a query is defined as [Van Rijsbergen 1979]:

$$\frac{1}{N}\sum_{i=0}^{N-1}\bar{\pi}\left(\frac{i}{N-1}\right) \tag{7-5}$$

Where

$$\bar{\pi}(x) = \max \pi_i$$
$$\frac{r_i}{r_n} = x \tag{7-6}$$

Where x is a recall point between [0 ...1] and $\pi(x)$ is precision at that point. Typically, 11-point interpolated average precision is used. This measure can be applied to individual queries, or averaged over several queries in a collection and expressed as a percentage.

Several other measures are used, also based on precision and recall [Buckley and Voorhees 2000]:

- $P(N_R)$ - the *precision* after the first N_R documents are retrieved, where N_R is the number of relevant documents for a query.
- $R(N_R)$ - recall after N_R documents are retrieved.
- Means Average Precision - mean (non-interpolated) average precision.
- Precision at 0.5 Recall - Precision after half the relevant document have been retrieved.

Buckley and Voorhees [2000] suggest that a cut-off level of 10 to 20 is acceptable for the evaluation of Web search-engines. A cut-off level is a rank that defines the retrieval set, for example, a cut-off level of 10 specifies that the top 10 documents build the retrieval set (and therefore the basis for the evaluation measurement).

7.2.2. Test Collections

A test collection for retrieval experiments contains three major components: a collection of documents, a set of queries, and an enumeration of the documents in the collection that are relevant to each query. The relevant documents are determined manually, and provide the standard against which all retrieval systems are measured. The optimal retrieval system, if it existed, would rank all documents designated as relevant higher

than those designated as non-relevant for all queries. The queries are designed to simulate questions that might be posed to the system in a realistic setting or, they may represent actual user queries [Witten et al., 2000].

A large number of test collections have been developed over the years for use in retrieval experiments. They differ very widely in terms of size and content. The best choice of retrieval strategy will depend, at least to some extent, on the characteristics of the collection, so the goal will be to find a method that performs well in a wide variety of situations. Among the important properties that vary between tests collections are:

1) number of documents
2) style and content of collection
3) length and specificity of query
4) number of relevant documents
5) length of documents

The size of the collection is important for measuring the performance of a system. Computationally intensive methods may perform well for small collections but are of limited use on very large collections. Systems are often refined by adding a screening step to get rid of the documents that are most clearly not relevant. Collection size also plays a role in issues, such as document representation, indexing, and document scoring algorithms.

The distribution of words over documents depends heavily on the style and content of the collection. Some collections, such as, news articles will contain a large proportion of proper names which could be very useful if recognized. Words that are rare in general use may be commonly used in a homogenous collection and vice versa. Documents may cluster based on common stylistic queues rather than similarity of content, which can make it harder to use relevance feedback on heterogeneous collections. A system that is optimized for a particular collection may fail to perform as well on another collection with a different word-document distribution. Thus, the best retrieval strategy may depend greatly on the length and specificity of the query. Complex data-driven retrieval strategies may have little success with short queries and limited amounts of information. However, the detail that they provide may be vital to obtain good results for longer, more precisely defined queries where little vocabulary is shared by relevant documents,

so that that the system may be required to have some language understanding capability in order to discover relevant documents.

The number of relevant documents does not influence the evaluation of an ad-hoc query, but is very important for routing (the dissemination of incoming documents to appropriate users on the basis of user interest profiles) or relevance feedback. Even the best relevance feedback strategy may not be particularly helpful if there are very few relevant documents to discover. The number of relevant documents (number of observations) is particularly important when using statistical classification for the routing problem [Baeza and Ribeiro-Neto 1999].

Document length can be an important issue in the similarity ranking of documents. Clearly, if documents have widely different lengths, then it is important to have a reasonable document normalization strategy. In addition, long relevant documents may be diluted with large quantities of unrelated information, and thus receive a much lower similarity score. Shorter documents may display higher variability in their similarity scores and thus some will tend to be ranked higher than longer documents.

7.3. Performance Evaluation

This research proposed an evolutionary approach for solving an information retrieval problem. The problem is, ability to retrieve from database only those documents that are relevant to a user's information needs, excluding documents that are irrelevant. This Chapter is devoted to testing and evaluation of the developed system. The system was tested for two different learning techniques, namely, relevance feedback (RF) adopting vector space model [Salton and Wong 1997] and interactive evolutionary reinforcement (IER) , which is a combination of relevance feedback and genetic algorithms. In both techniques fuzzy logic has been utilised for interactive user feedback.

7.3.1. Experimental Methodology

The efficiency of the IER technique is determined in terms of two performance measures: learning and retrieval performance. *Learning performance* involves carrying out many iterations with the same data set, in order to fix the GA parameters. To conduct this evaluation one query is used for 10 generations. For each experiment, different GA parameters are selected. After determining the optimal GA parameters, the

precision and recall were measured to evaluate *the retrieval performance* of the system. The IRS was tested for user-needs in ten different subject domains. Each query was designed to retrieve the 10 "best documents", which were assessed by an experienced user in each of the subject areas for 10 successive retrievals. Results incorporate three performance metrics: Recall, Precision and Cut-off (control point at which Recall and Precision measures are made).

Statistical significance tests between methods describe the superiority of one method over another. They are used to compare two retrieval methods and decide if any one produces measurably better results than the other. The most common approach to this problem is to apply a paired t-test [Hull 1993]. This test compares the magnitude of difference between methods to the variation among the difference. If average difference is large compared to its standard error, then the methods are significantly different.

Sparck-Jones [1974] and Buckley and Voorhees [2000] proposed another way to evaluate systems by comparison of two information retrieval methods. They suggested a fuzzy measure based on differences in scores between two runs. In their analysis of TREC data [Buckley and Voorhees 2000] findings show that the error rate decreases as the fuzziness increases. However, they also state that no firm conclusions can be drawn from the results, as most methods are considered equal.

The evaluation experiments were conducted with a collection of 10 queries (each query represents a user profile) and 400 documents. The system was tested with assistance of three PhD students knowledgeable of using different search engines. To satisfy the user-needs, it was essential to select a range of appropriate documents with which they were familiar and used frequently. The group of the people that were chosen to test the system, represented different fields of study including, system design, artificial intelligence, supply chain management and information retrieval. The 400 documents were selected as the top hits from Bath Information and Data Services (BIDS) [www.bids.co.uk] for the key words, "agent", "multi-agent", "information retrieval", "genetic algorithms", "fuzzy logic", "supply chain", "machine learning", "search engine", "e-commerce", "knowledge management" and "design methodology" (Table 7-3). The format of all files was plain ASCII and the relevant documents for each category were determined manually. These keywords represent the domain of knowledge of the system users (3 PhD students).

In these experiments, the measures of effectiveness were precision and recall. In order to conclude which methods perform better, the paired t-test significance test was used. The documents overlapped between different categories; for example, a document titled "Genetic Algorithms for Information Retrieval", could belong to both the GA and IR categories. A feedback form similar to Tables 7-1 and Table 7-2 were provided for each evaluation.

Query : ***Information Retrieval***

Method: IER

	Retrieved Documents									
Generations	1	2	3	4	5	6	7	8	9	10
1	Vp	vp	vp	m	g	p	p	m	e	m
2	Vp	vp	vp	p	g	p	p	m	e	p
3	Vp	p	vp	vp	p	m	e	m	vg	g
4	P	p	g	vp	g	p	vp	m	vg	m
5	P	p	vp	g	vg	m	m	g	g	p
6	Vp	vp	vg	e	vg	g	m	g	p	m
7	P	p	m	vg	e	g	m	g	g	m
8	M	p	vp	vg	vp	vg	vg	g	vp	m
9	P	p	vg	e	vg	vp	vg	m	g	g
10	G	vg	e	p	vg	m	m	m	g	m
11	M	p	g	g	vg	m	g	e	vg	g
12	M	m	g	g	vg	g	g	m	e	g
13	G	e	p	vg	p	p	m	p	p	p
14	Vp	vp	g	p	vg	p	m	m	p	vg
15	P	m	g	vg	g	p	m	p	p	p
16	M	m	m	p	vg	g	vg	m	p	p
17	G	g	g	p	m	vg	m	vg	p	p
18	Vp	p	p	e	vg	e	m	vg	e	m
19	G	m	e	p	m	g	p	g	vg	g
20	G	e	g	g	vg	g	g	e	e	vp

(e: excellent, vg: very good, g: good, m: moderate, p: poor, vp: very poor)

Table 7-1: Assessment form *IER* method

Query: information Filtering **Method : Relevance Feedback**

			Retrieved Documents							
Generations	1	2	3	4	5	6	7	8	9	10
1	p	p	p	p	vp	p	p	p	vp	p
2	vp	vp	p	p	p	p	p	g	p	p
3	p	g	vp	vp	vp	vp	vp	g	vp	m
4	vp	p	vp	vp	vp	vp	vg	p	vg	vp
5	p	vp	vp	vp	g	vp	vp	g	vg	vp
6	m	p	vp	vp	m	vp	vp	vp	m	e
7	m	vp	vp	vg	vp	vp	e	vp	vg	g
8	m	vp	vp	vg	vp	vg	e	vp	vp	g
9	m	vp	vp	vg	vp	e	e	vp	vp	g
10	vp	vp	vp	vg	vp	vp	e	e	g	g
11	vp	vp	vp	vp	vp	e	e	e	g	g
12	vp	vp	vp	vp	vp	e	e	e	g	g
13	vp	vp	vp	vp	vp	e	e	e	g	g
14	vp	vp	vp	vp	vp	e	e	e	g	g
15	vp	vp	vp	vp	vp	e	e	e	g	g
16	vp	vp	vp	vp	vp	e	e	e	g	g
17	vp	vp	vp	vp	vp	e	e	e	g	g
18	vp	vp	vp	vp	vp	e	e	e	g	g
19	vp	vp	vp	vp	vp	e	e	e	g	g
20	vp	vp	vp	vp	vp	e	e	e	g	g

(e: excellent, vg: very good, g: good, m: moderate, p: poor, vp: very poor)

Table 7-2: Assessment form relevance feedback method.

The other documents in the collection were selected from other areas which can have similar words, for example, "estate agent" contains the word, "agent", which can apply to a software agent as well. A summary of the document collection is given in Table 7-3.

Subject areas	Number of relevant document in the collection
Information retrieval/ filtering	60
Genetic algorithms	16
Fuzzy Logic	26
E-commerce	24
Design methodology	16
Search engine	34
Agent / multi-agent	34
Supply chain	20
Machine Learning	30
Knowledge Management	32
Others	108
Total	400

Table 7-3: Documents collection

7.3.2. Learning Performance Result

In order to measure learning performance, only GA crossover probability (p_c) was tuned. Due to the lesser importance of mutation, the probability of mutation(p_m) was set as a standard value of 0.01. As explained above, each search produced a list of ten documents and, the retrieval was repeated for 10 generations. Table 7-4 illustrates the retrieval process for one search query. Each precision result is a mean of four search results.

Query = Information Retrieval					Pc Value			
	Generation	0.45	0.55	0.65	0.7	0.75	0.8	0.85
	1	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	2	0.2	0.2	0.3	0.3	0.1	0.3	0
	3	0.1	0.4	0.5	0.6	0.2	0.4	0
	4	0.2	0.4	0.7	0.6	0.5	0.9	0.1
	5	0.3	0.5	0.7	0.8	0.6	0.8	0.1
	6	0.2	0.6	0.7	0.8	0.7	0.6	0.3
	7	0.3	0.6	0.8	0.9	0.7	0.7	0
	8	0.2	0.4	0.5	0.7	0.7	0.3	0.1
	9	0.4	0.6	0.9	0.9	0.2	0.4	0.1
	10	0.5	0.6	1	1	0.3	0.3	0.1
	Average	0.26	0.45	0.63	0.68	0.42	0.49	0.1

Table 7-4: Precision results for learning performance

As shown in Table 7-4, low values for the crossover parameter do not show significant learning improvement. This is because at low crossover values, fewer new individuals are introduced to the population, which results in a longer time for the system to improve its performance. Conversely, too high a value for the crossover parameter results in introducing new individuals too quickly into the population. This causes the system to change the population of user models more quickly and more randomly, regardless of user feedback. The best performance of the system is given by medium values (0.65 ... 0.7) of crossover probability. As shown in Figure 7-5 the system is more stable for crossover parameter value of 0.7 and, the number of relevant documents in each search result is also considerably higher than for the other parameter values. Also considering Figure 7-6, which shows the total relevant documents retrieved in the 10 searches, the crossover probability of 0.7 has the highest number of relevant documents.

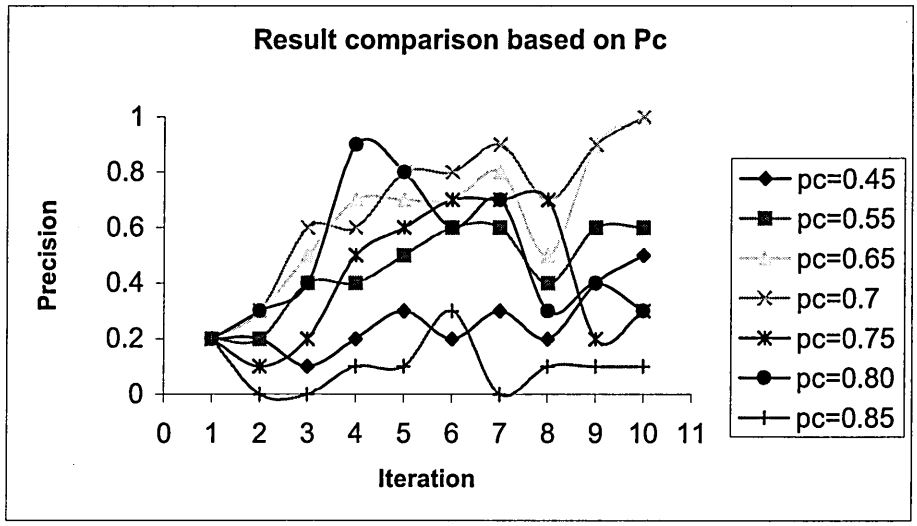


Figure 7-5: Precision Graph for crossover parameters with all values.

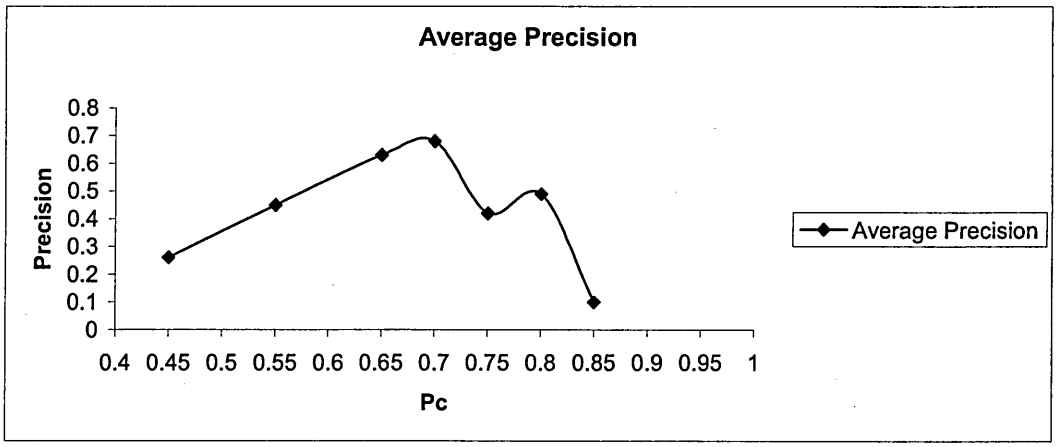


Figure 7-6: Average precision for each parameter.

7.3.3. Retrieval Results

In this work, a comparison was made between the retrieval performance of conventional relevance feedback and an IER method based on a classical GA and an interactive human relevance feedback. Both techniques were tested under the same experimental conditions. New measurements were proposed named, weighted precision and weighted recall which

are similar to Mea's new effectiveness measurement [Mea and Mizzaro 2004] . The standard methods for calculating precision and recall are based on a binary measure of relevance, while in the proposed system graded linguistic scores are used:

$$weighted\ \mathrm{Re}call = \frac{\sum_{i=1}^{10} w_i}{r} \tag{7-8}$$

$$weighted\ \mathrm{Pr}ecision = \frac{\sum_{i=1}^{10} w_i}{n} \tag{7-9}$$

From equations 7-8 and 7-9, it can be shown that,

$$weighted\ \mathrm{Pr}ecision = \frac{weighted\ \mathrm{Re}call * r}{n} \tag{7-10}$$

Where w is relevance judgement value (Table 7-5), n is the total number of retrieved documents and r is the number of relevant items in the collection. In general, there is a problem knowing how many relevant documents there are in a collection, especially for online database. In our test, however, r is known. Table 7-5 shows the weighting of the user relevance feedback. In the rest of the thesis, the terms precision and recall are used for weighted precision and recall.

Relevance judgment	w
Excellent	1.0
Very good	0.8
Good	0.6
Moderate	0.4
Poor	0.2
Very poor	0.0

Table 7-5: Feedback weight values

Retrieval effectiveness was demonstrated through a recall-precision graph. The graph was constructed from the ranked list of components retrieved for the user (query table: list of subject area in data collection). For the purpose of comparison, recall and precision graphs were constructed for the two different IR methods, using *cut-off of 10.*

Tables 7-6, Table 7-7, Figures 7-7 and 7-8 show the recall and precision result for 20 generations for both IER and RF. According to the known number of relevant documents in the collection, the maximum value for precision is (10/10=1) and maximum value for recall is (10/16 = 0.65).

Generation	recall	precision	average precision	Average recall
1	0.07	0.39	0.39	0.07
2	0.05	0.31	0.35	0.06
3	0.07	0.42	0.39	0.06
4	0.07	0.41	0.40	0.07
5	0.08	0.48	0.44	0.07
6	0.09	0.56	0.50	0.08
7	0.10	0.62	0.56	0.09
8	0.08	0.48	0.52	0.09
9	0.10	0.61	0.56	0.09
10	0.11	0.68	0.62	0.10
11	0.12	0.70	0.66	0.11
12	0.12	0.72	0.69	0.12
13	0.07	0.44	0.57	0.09
14	0.07	0.43	0.50	0.08
15	0.08	0.45	0.47	0.08
16	0.09	0.55	0.51	0.09
17	0.10	0.57	0.54	0.09
18	0.11	0.64	0.59	0.10
19	0.11	0.63	0.61	0.10
20	0.12	0.74	0.68	0.11

Table 7-6: Precision and recall values for IER with query: Information Retrieval

Generation	recall	precision	average precision	average recall
1	0.03	0.16	0.16	0.03
2	0.04	0.21	0.19	0.03
3	0.04	0.22	0.20	0.03
4	0.04	0.22	0.21	0.04
5	0.04	0.25	0.23	0.04
6	0.05	0.30	0.27	0.04
7	0.07	0.41	0.34	0.06
8	0.07	0.41	0.37	0.06
9	0.07	0.42	0.40	0.07
10	0.07	0.43	0.41	0.07
11	0.07	0.44	0.43	0.07
12	0.07	0.44	0.43	0.07
13	0.07	0.44	0.44	0.07
14	0.07	0.44	0.44	0.07
15	0.07	0.44	0.44	0.07
16	0.07	0.44	0.44	0.07
17	0.07	0.44	0.44	0.07
18	0.07	0.44	0.44	0.07
19	0.07	0.44	0.44	0.07
20	0.07	0.44	0.44	0.07

Table 7-7: Precision and recall values for RF with query: Information Retrieval

Figure 7-7 and 7-8 graphically illustrate the comparison precision and recall between IER against RF.

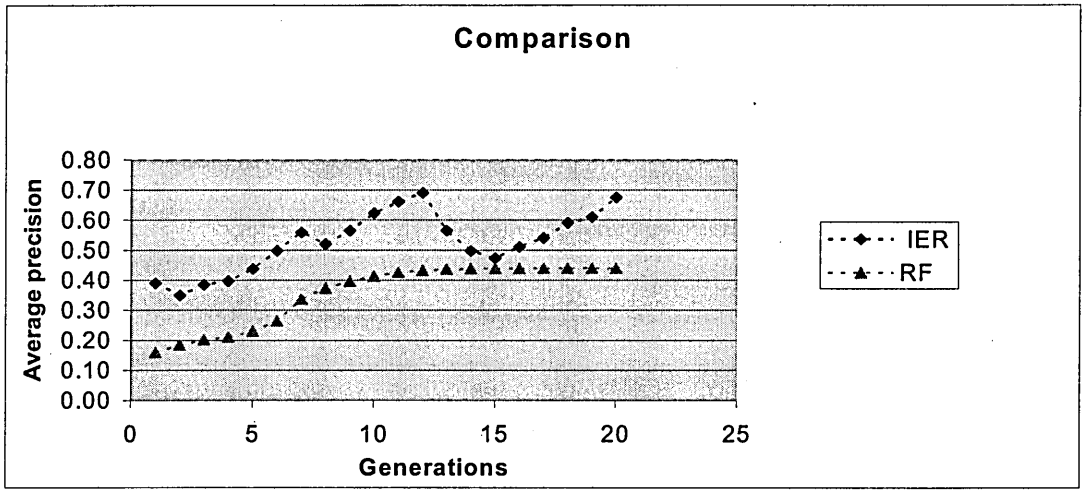


Figure 7-7: Average Precision for Query: Information Retrieval

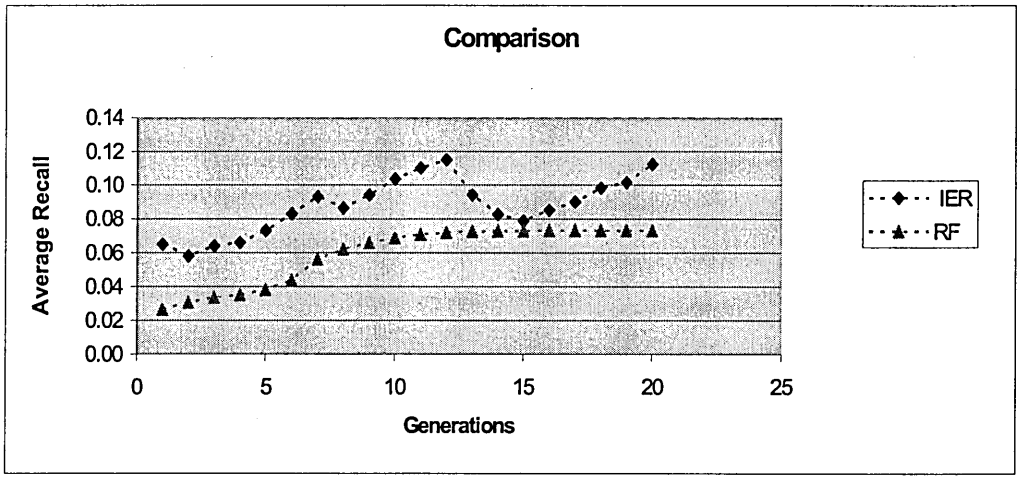


Figure 7-8: Average Recall for Query: Information Retrieval

From the Figures, it is shown that the average recall for IER is higher than for RF. The average precision for IER is also higher than in RF. However, it is also observed that in the case of IER there are sudden surges in precision over the generations, compared to RF. This can be attributed, mainly, to the effect of crossover and mutation in GA. It is, however,

important to note that the precision is still usually higher than with RF. Similar test results for other queries are shown Figures 7-9 to 7-26. In some test cases, RF is shown to perform better than IER. One possible explanation is that the pool of terms is not sufficient to cover all the relevant terms, and also the number of relevant documents for these tests is much smaller than the other areas. Figure 7-27 and 7-28 show the average precision and recall over all queries, respectively. Figures 7-29 and 7-30 show the precision-recall graphs for IER and RF, respectively. While, it is apparent that IER, in general, performs better than RF, a statistical analysis was carried out to determine the significance of any difference between the two methods.

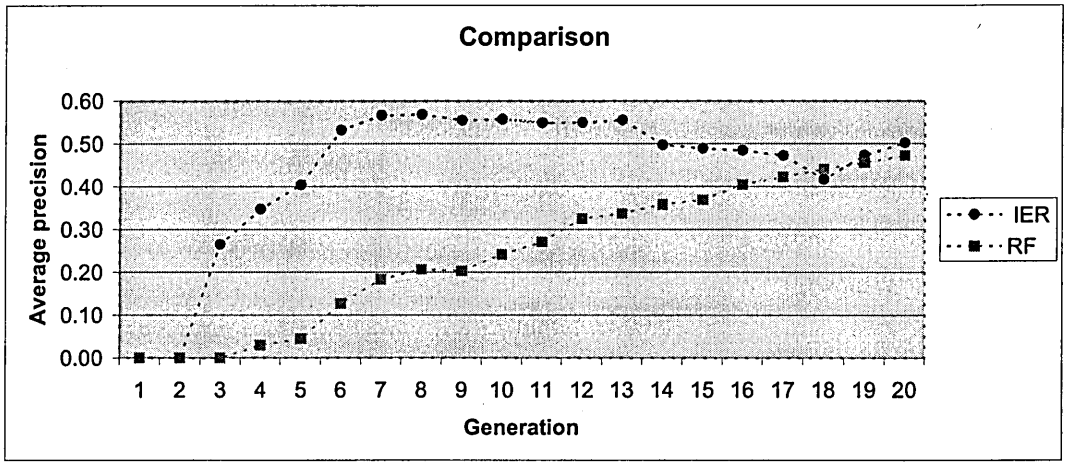


Figure 7-9: Average Precision for Query: Agent

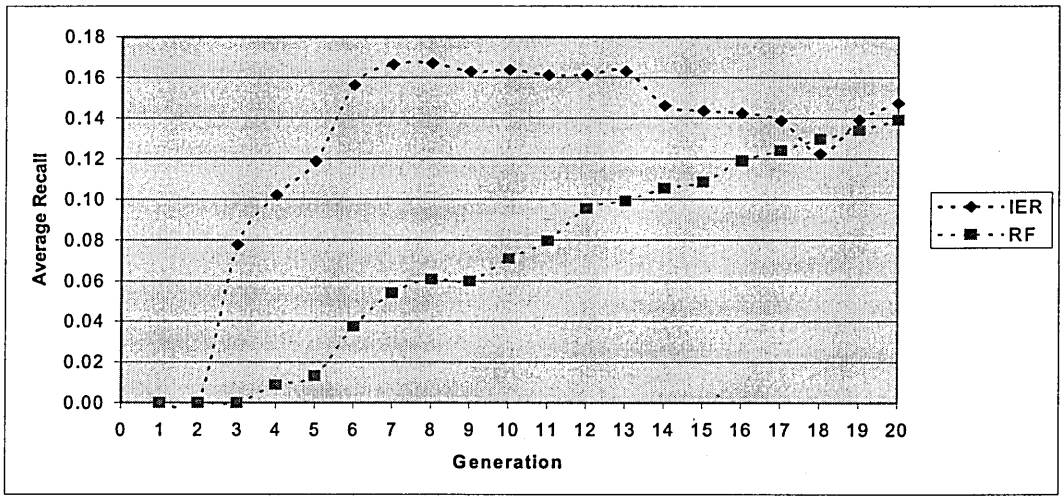


Figure 7-10: Average Recall for Query: Agent

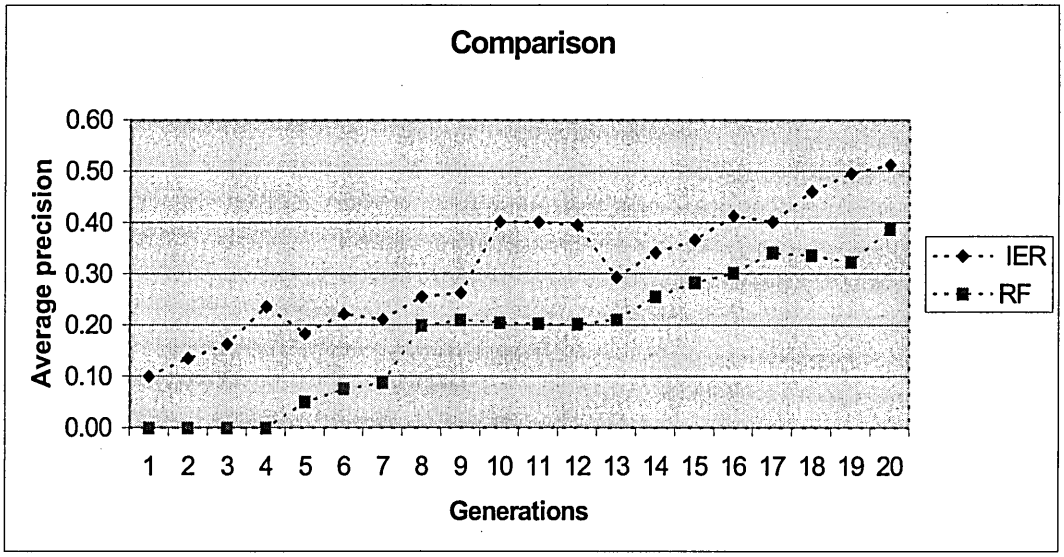


Figure 7-11: Average Precision for Query: Fuzzy

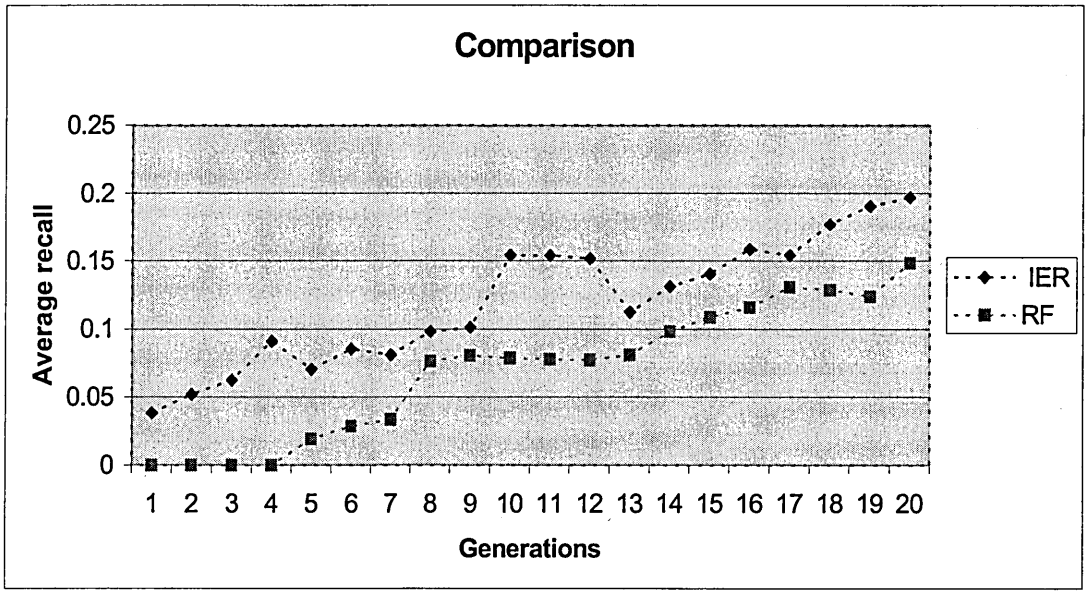


Figure 7-12: Average Recall for Query: Fuzzy

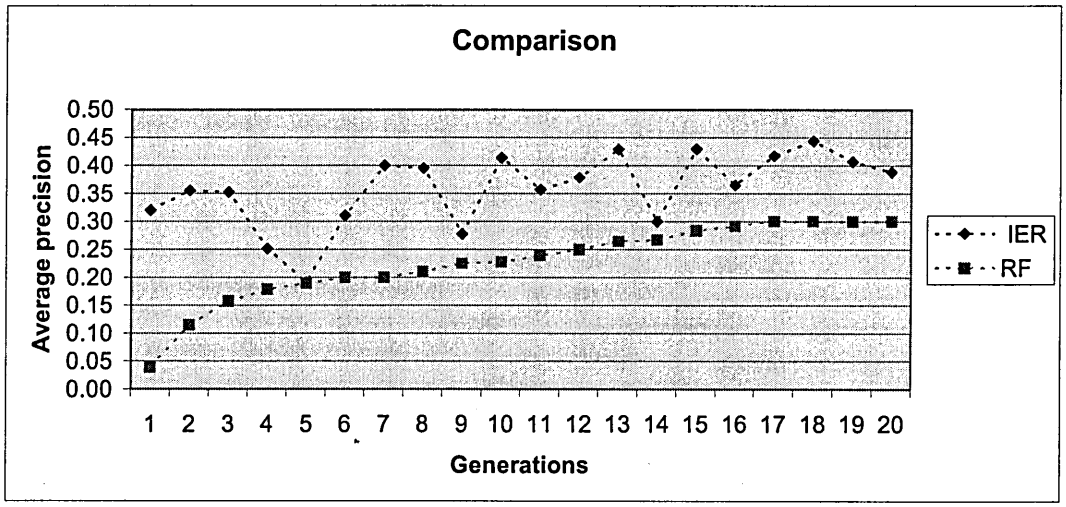


Figure 7-13: Average Precision for Query Search: Engine

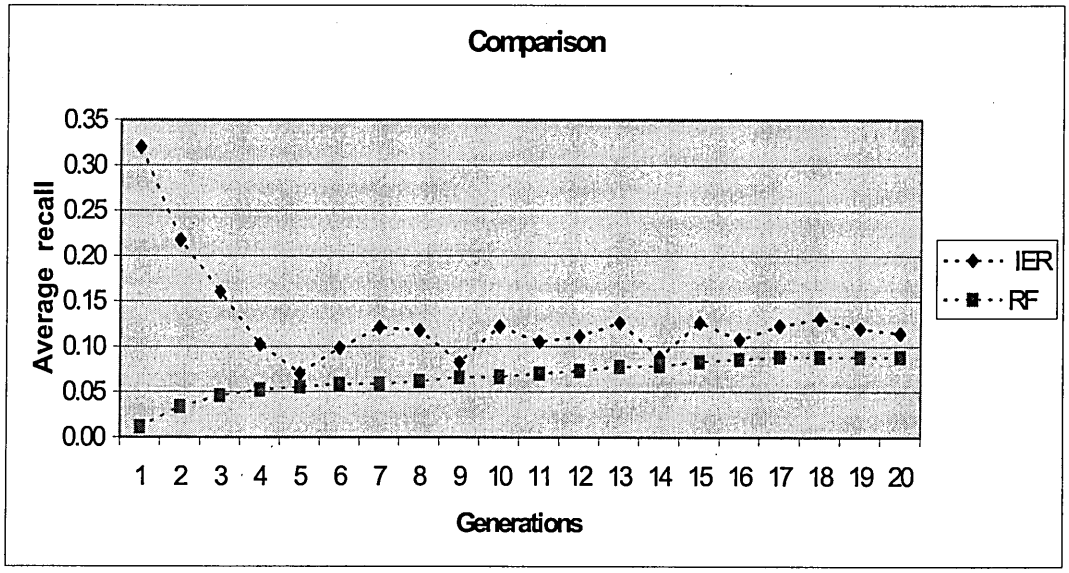


Figure 7-14: Average Recall for Query Search: Engine

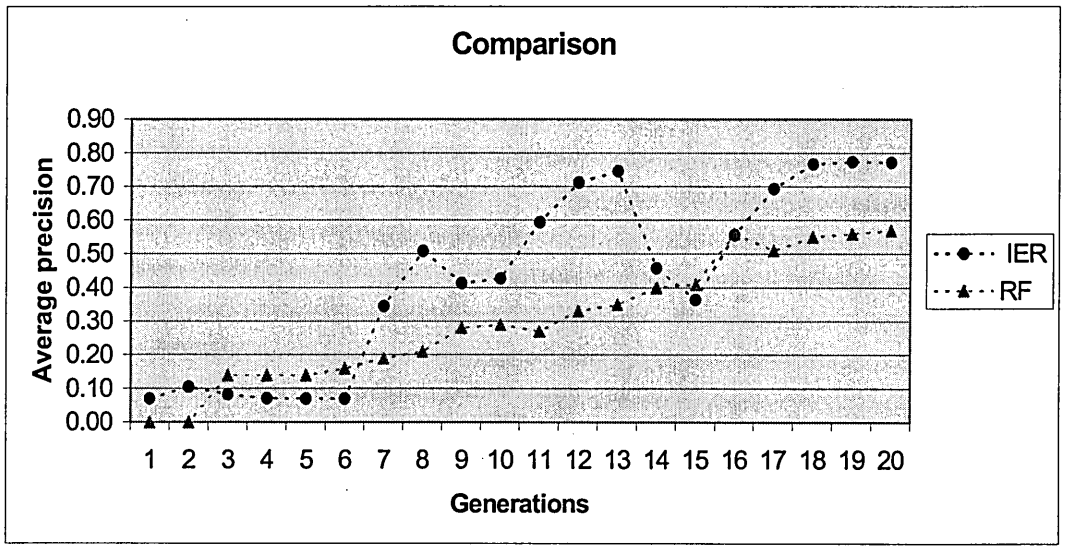


Figure 7-15: Average Precision for Query: Machine Learning

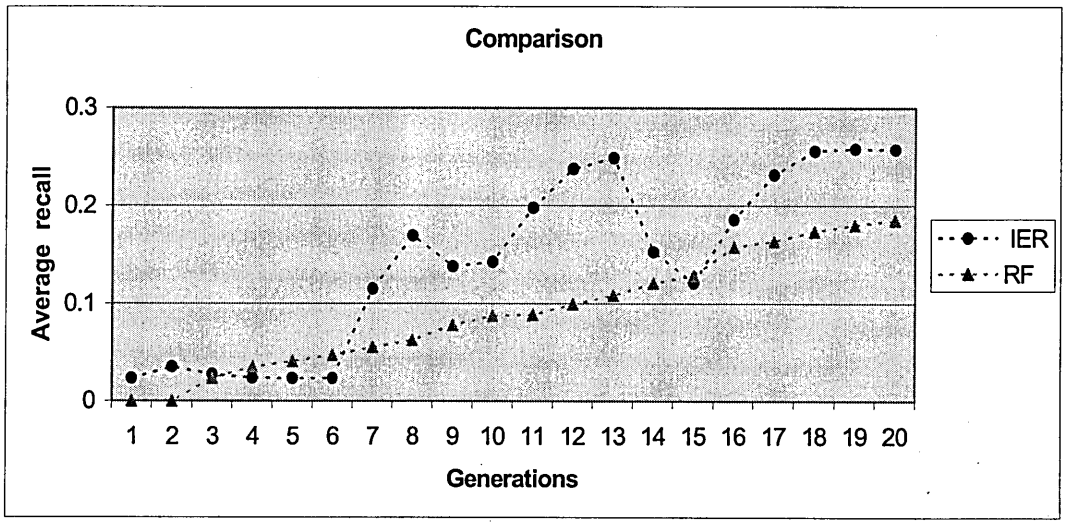


Figure 7-16: Average Recall for Query Machine Learning

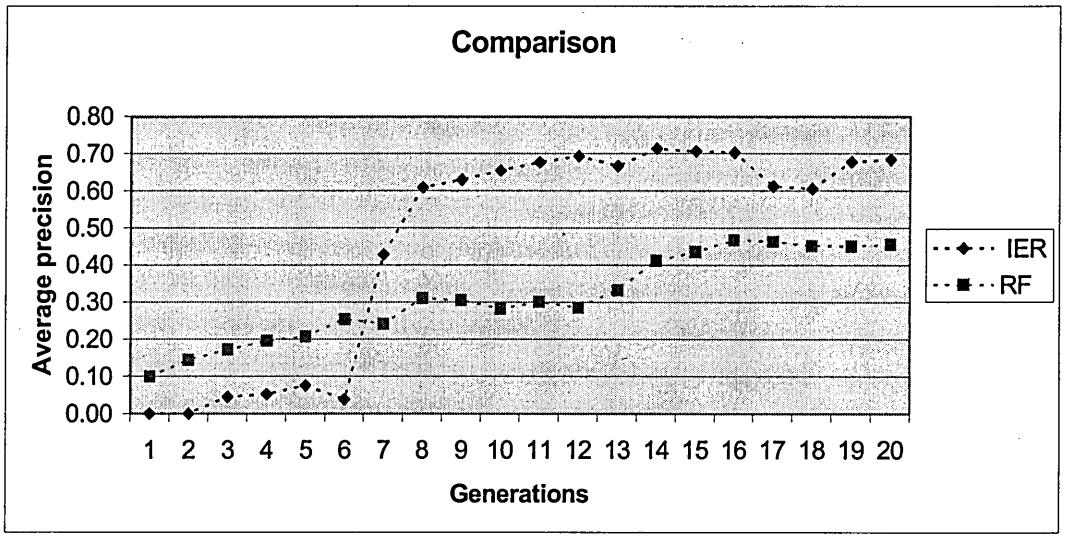


Figure 7-17: Average Precision for Query: Knowledge Management

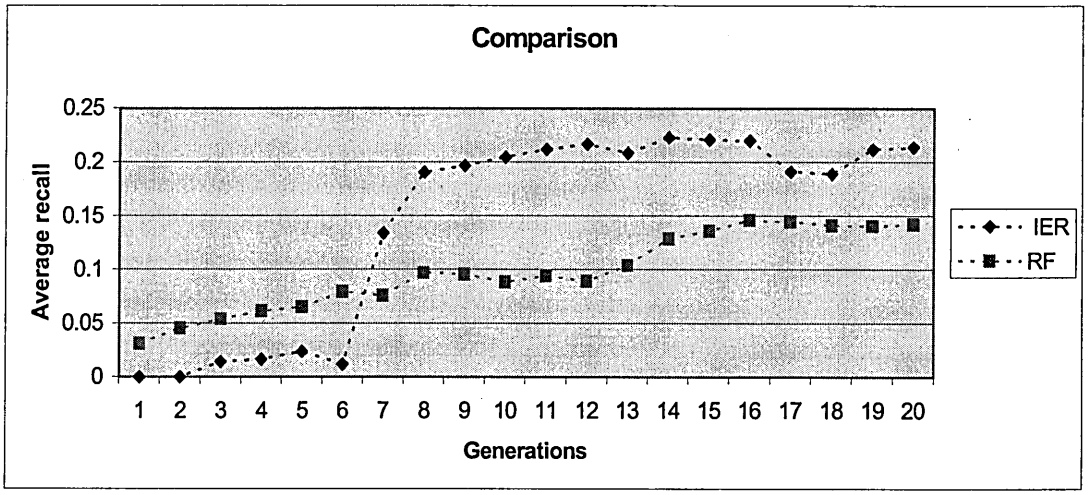


Figure 7-18: Average Recall for Query: Knowledge Management

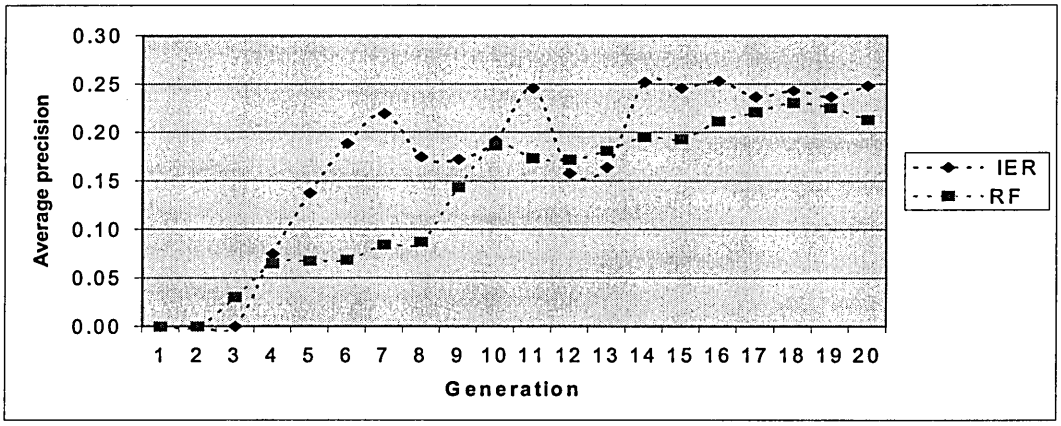


Figure 7-19: Average Precision for Query: Genetic Algorithms

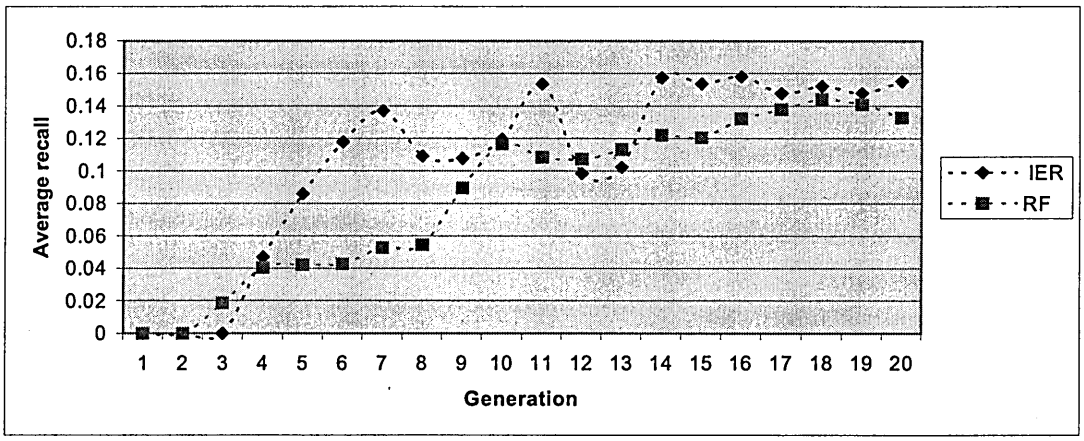


Figure 7-20: Average Recall for Query: Genetic Algorithms

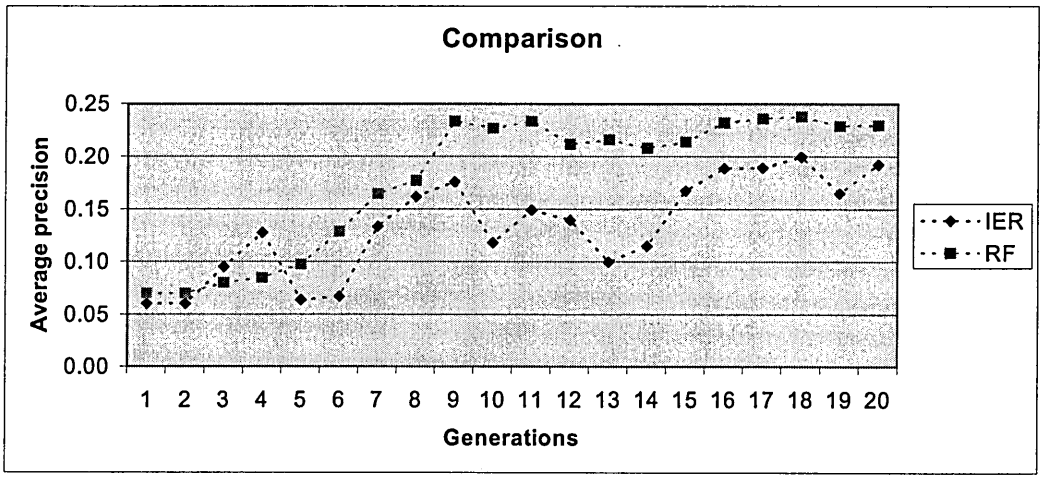


Figure 7-21: Average Precision for Query: Design Method

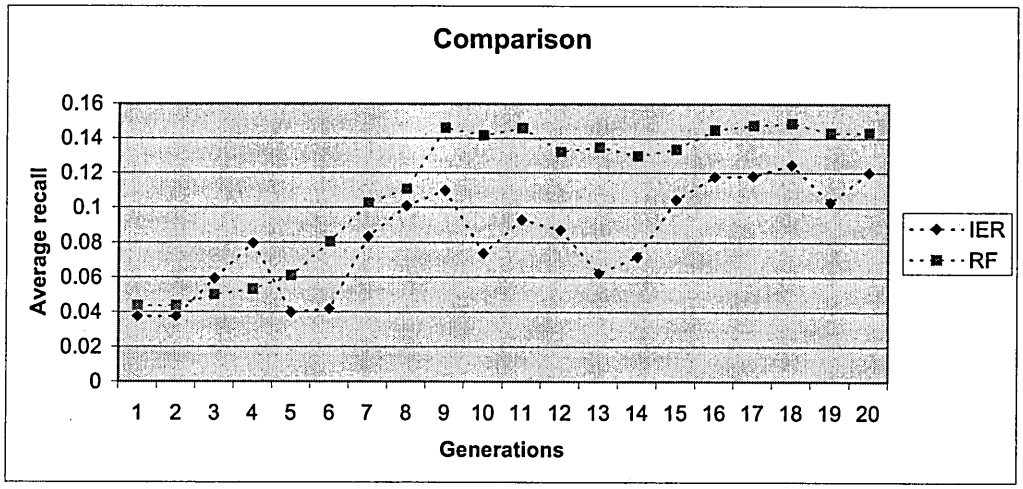


Figure 7-22: Average Recall for Query: Design Method

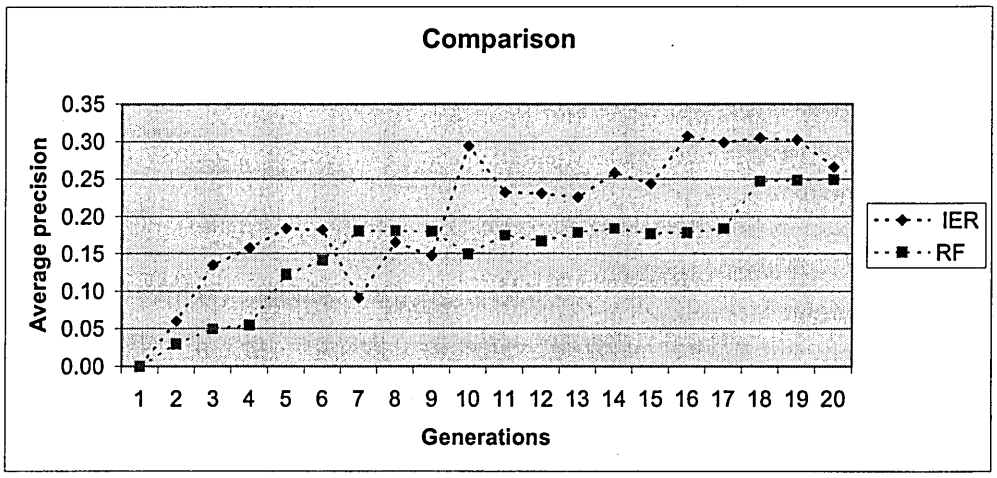


Figure 7-23: Average Precision for Query: E-commerce

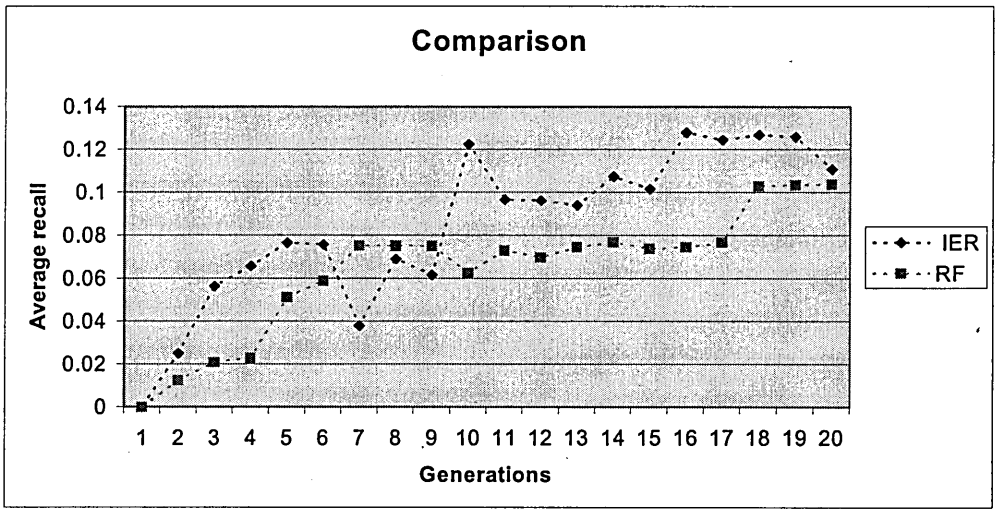


Figure 7-24: Average Recall for Query: E-commerce

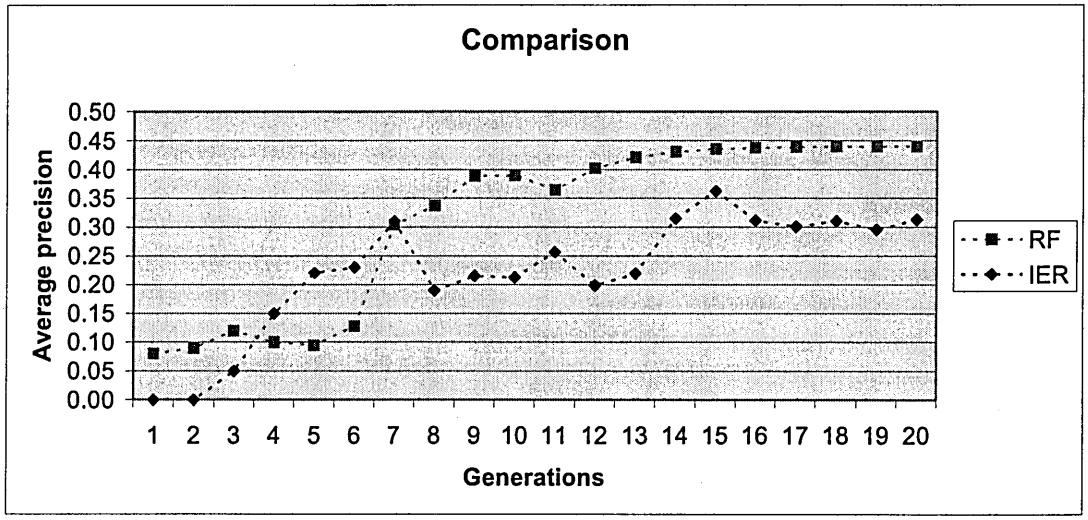


Figure 7-25: Average Precision for Query: Supply Chain

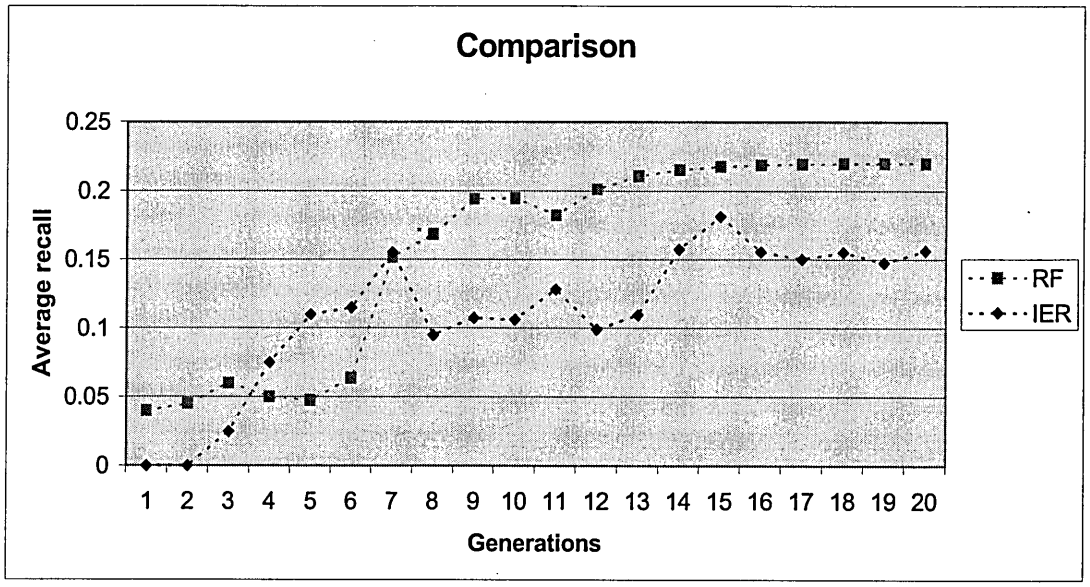


Figure 7-26: Average Recall for Query: Supply Chain

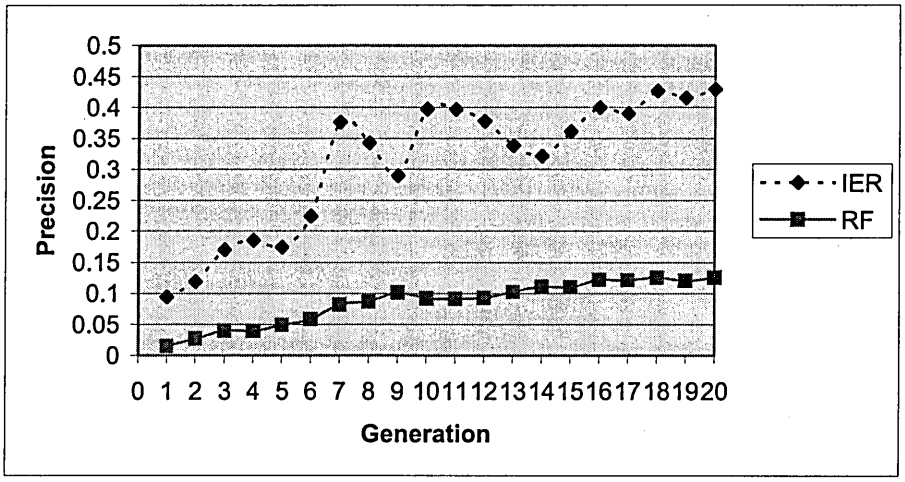


Figure 7-27: Average Precision for all Queries

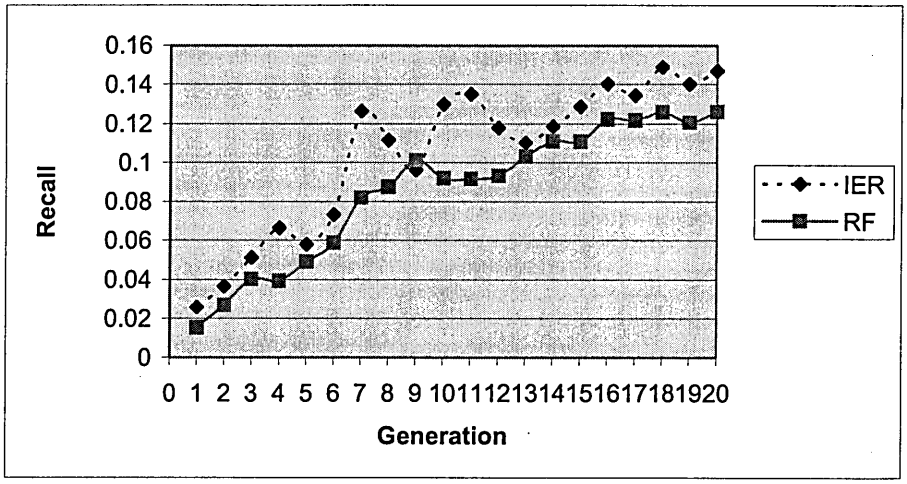


Figure 7-28: Average Recall for all Queries

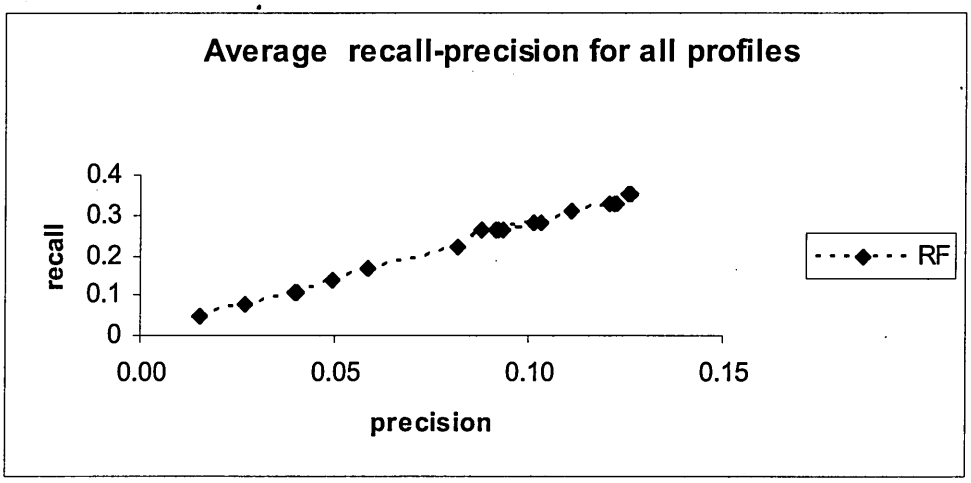


Figure 7-29: Precision-recall graph for RF technique

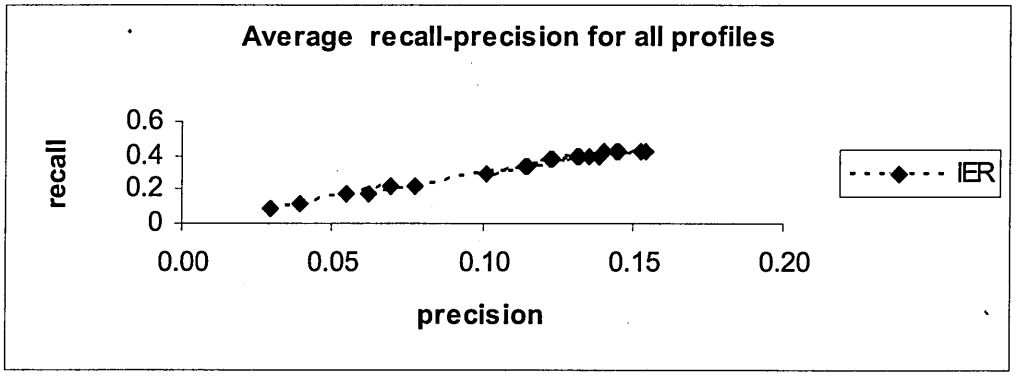


Figure 7-30: Precision-recall graph for IER technique

7.3.4. Statistical Analysis

Two statistical analyses were carried out, namely, significance test and correlation. The significance test measures the effectiveness of IER in comparison to RF. Another study was performed to determine any correlation between the number of relevant documents in a document collection and the resultant retrieval precision.

Significance test

The aim of this test is to determine any improvements of one method over another method. Thus, a significance test was adopted to reject the null hypothesis, H_0 that there is no difference between method IER and RF. The idea is to show that, on the basis of result data, the null hypothesis is indefensible, because it is associated with an implausible low probability. Rejecting H_0 implies accepting the alternative hypothesis, H_1 that either method IER consistently outperforms method RF, or vice versa.

H_0: average precision IER – average precision RF <= 0

H_1: average precision IER >average precision RF

The hypothesis was tested by comparing average precision values, across all queries, after 20 generations.

Let X_i *and* Y_i be the scores of retrieval methods X and Y for query i where $i=1....n$, and define $D_i = Y_i - X_i$.

The paired t-test is given by,

$$t = \frac{\overline{D}}{s(D_i)/\sqrt{n}} \tag{7-7}$$

with

$$\overline{D} = \frac{1}{n}\sum_{i=1}^{n} D_i$$

and

$$s(D_i) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(D_i - \overline{D})^2}$$

Query	Average precision IER	Average precision RF
IF & IR	√ 0.68	0.44
GA	√ 0.25	0.21
Fuzzy	√ 0.51	0.39
ML	√ 0.77	0.56
E-commerce	√ 0.27	0.25
Design method	0.19	0.23
Search engine	√ 0.39	0.30
Agent/MA	√ 0.5	0.47
Supply chain	0.31	0.44
Knowledge management	√ 0.68	0.46

Table 7-8: Average Precision for Queries

The paired t-test is, normally, used for small samples with normal distribution. The result of the test, from the data in Table 7-8, was t =2.10 at 0.05 significance (Figure 7-31). The interpretation of this result is that the performance of IER is statistically better than RF.

Paired T-Test and Confidence Interval

```
Paired T for Avg Precision IER - Avg Precision RF
                  N      Mean     StDev   SE Mean
Avg Prec         10    0.4550    0.2047    0.0647
Avg Prec         10    0.3750    0.1196    0.0378
Difference       10    0.0800    0.1202    0.0380
95% CI for mean difference: (-0.0060, 0.1660)
T-Test of mean difference = 0 (vs > 0): T-Value = 2.10
P-Value = 0.032
```

Figure 7-31: t-test result

Correlation

Another statistical analysis was to measure the degree of association between average precision and number of relevant document in the collection in each search category. The reason for only considering precision is based on the definition of recall, which if there were more relevant documents in the collection there would be lower recall, and visa versa.

The data for analysis is shown in Tables 7-9 and 7-10. A scatter diagram of the average precision against the number of relevant documents in the collection is shown in Figure 7-32; showing no obviously significant difference in the patterns.

Subject	Number of Document	Average precision
IF & IR	60	0.44
GA	16	0.21
Fuzzy	26	0.39
ML	30	0.56
E-commerce	24	0.25
Design method	16	0.23
Search engine	34	0.30
Agent/MA	34	0.47
Supply chain	20	0.44
Knowledge management	32	0.46

Table 7-9: Number of documents against retrieval performance in RF method

Subject	Number Of Documents	Average Precision
IF & IR	60	0.68
GA	16	0.25
Fuzzy	26	0.51
ML	30	0.77
E-commerce	24	0.27
Design method	16	0.19
Search engine	34	0.39
Agent/MA	34	0.50
Supply chain	20	0.31
Knowledge management	32	0.68

Table 7-10: Number of documents against retrieval performance in IER method

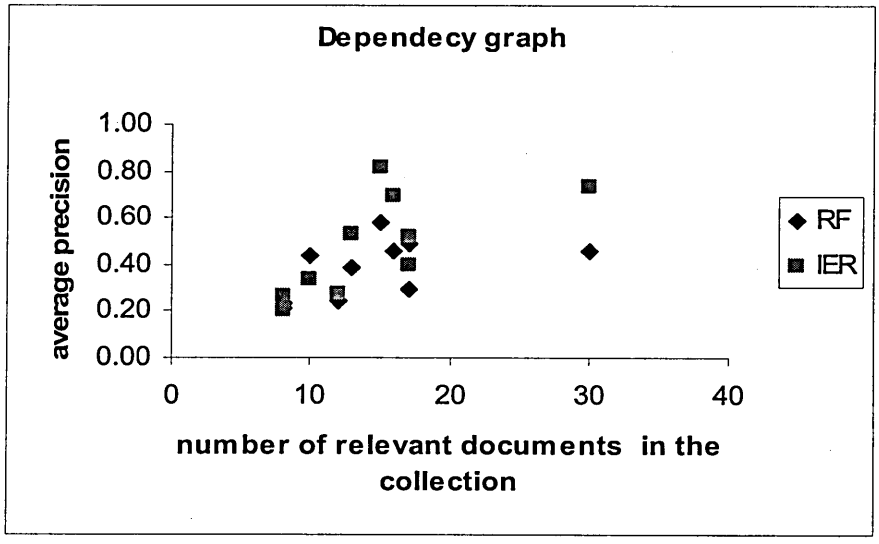


Figure 7-32: Average precision and number of document

COREL function in Microsoft Excel was used to determine the correlation coefficients between precision and the number of relevant documents. Using the data in Tables 7-9

and 7-10, a correlation of 0.47 was determined for RF and 0.69 for IER. According to Cohen and Holliday (1982), correlation values below 0.2 indicate an unlikely correlation; values between 0.2 and 0.4 weak correlation; between 0.4 and 0.6, a moderate correlation; 0.6 to 0.8, a strong correlation, while values above 0.8 indicate a very strong correlation. Thus, there is evidence that there is a moderate correlation between average precision and the number of relevant documents using RF and a strong correlation using IER.

7.3.5. Discussion

Results obtained in this study would suggest that IER, in general, performs better than RF. This result seems to contradict result reported in previous studies using evolutionary learning, such as Vrajitoru [1998] and Chen and Dhar [1998]. However, there is a support for the superiority in performance of IER. IER uses on-line reinforcement learning through interaction with human users. Human interactive reinforcement provides a direct evaluation of the relevance of documents, namely, *user preference* that cannot be expressed by any analytical fitness function. This results in a user model that is, in fact, optimised by the user. The encoding technique and the operators for the GA used with IER is also different from previous studies. The chromosomes are encoded by terms and their weights, as opposed to terms only [Vrajitoru 1998]. The results also indicated that, in most cases, the maximum value for precision was reached in about 10 generations, which suggests that learning can achieved in a relatively short period of interaction.

The standard shape for the precision-recall graph [Keen 1997] (Figure 7-4) describes an inverse relationship between recall and precision, whereas the result in this study suggests a linear relationship, similar to [Chen and Kuo 2000] work. However, as discussed earlier (equation 7-10), this in fact is the case when the number of documents retrieved and number of relevant document in the collection are fixed. Jansen et al. [1998] research on real- life information retrieval, revealed that over half of users did not access results beyond the top ten ranked documents. Thus, the rationale for constraining the number of retrieved is pragmatic and justified.

7.4. Summary

This Chapter discussed evaluation of the developed IRS. Firstly, the criteria used for evaluating the performance of IRS were presented. The evaluation focused on measurement of the retrieval effectiveness, namely, weightedPrecision and weightedRecall. The methodology for evaluation included a comparison of the new machine learning algorithm against conventional relevance feedback. Tests on the machine learning algorithm were its training and operational performances. In particular, training of the system could be achieved within a dozen sessions of interaction with user, while operational performance was, in general, superior to conventional relevance feedback. To verify the experimental evaluation results statistical analysis was carried out. The first test sought to establish the significance of the difference between the retrieval performances of the two approaches. Using a t-test analysis the developed algorithm was shown to be better than conventional relevance feedback to a confidence level of 95%. The evaluation was carried out under controlled conditions whereby the number of relevant and irrelevant documents in the test collection was known. Therefore, the second statistical test was to determine any correlation between the number of relevant documents in the test collection and the retrieval precision. Not surprising results show a weak to medium correlation for both approaches. The conclusion, on the basis of the evaluation tests, is that the proposed interactive evolutionary reinforcement provides an improved retrieval performance as compared to conventional relevance feedback.

Chapter 8. CONCLUSION

8.1. Overview

With information retrieval systems, bridging the gap between the physical characteristics of data with the user perceptions is challenging. In order to address this challenge, employing user profiles to improve the retrieval accuracy becomes essential. However, the system performance may degrade due to inaccuracy of user profiles. Therefore, for an approach to be effective, it should offer a learning mechanism to involve user input (feedback). Focusing on a text retrieval application, we proposed the adaptive information retrieval system to improve the profiles automatically using GA-based learning mechanism.

The main objective of this research set in chapter one are met and discussed in chapter one and chapter five. The main requirements for development of the adaptive information retrieval system categorized into two areas, functional and non-functional requirements.

To satisfy the functional requirements of an adaptive information retrieval system, an IRS was proposed, which is able to specialize to user interests, and explore new domains for potentially relevant information that can not be explored via a conventional IR process. This thesis has presented a novel approach for improving document retrieval efficiency through, so called, interactive evolutionary reinforcement. The thesis made contribution to the field of information retrieval, by combining fuzzy relevance feedback and genetic algorithms evolution, models of user information-needs which can be derived that optimise retrieval precision. The approach was validated on applications where user information needs are subjective but relatively static, and hence can be accurately modelled. This includes special-interest-group information retrieval, such as, retrieval of academic and research documents.

The machine learning algorithm was proposed to be incorporated into the developed IRS that would autonomously acquire knowledge about user information-needs, through interaction with the user. This approach requires less effort from the user, compared to interactive query mechanisms proposed in other machine learning approaches. The

system not only evolves subjective information relevance models, but also continuously adapts to cope with changing user interests. Results show that significant improvement in precision can be achieved over the use of conventional relevance feedback alone.

The functional requirements of specification for the proposed system revealed intractable complexities within the IRS. The basic processes involved in information retrieval include, document representation, user information-needs representation, comparison of documents against information-needs, and adaptation of information-needs models. To meet the requirements Novel GA operators (crossover and mutation) and a new method for chromosome encoding have been implemented. The work was complemented with additional use of the vector space model for document representation which was generalized for use with documents containing textual information. The distance metric for computation of document similarities and the effect of relevance feedback has also been generalized.

In order to satisfy the non-functional requirements of the proposed system, a multi-agent paradigm was used, which offers flexibility in system development, maintenance and can improve processing times. In previous multi-agent approaches for developing IRS, "agents" are complete information retrieval system, which are administered and coordinated by another agent. The system development methodology was based on recursive analysis of both IRS requirements specification and the agent development environments. As a part of this study an objective evaluation of current agent developments environments was conducted, which is believed to be unique.

The system performance was evaluated to determine adaptation to 10 different user profiles. For each, profile it was shown that the maximum retrieval precision was achieved in a relatively short period of interaction with the system. Furthermore, the retrieval precision was superior to that achieved by conventional relevance feedback. Both these results can be attributed to the user model learning method, namely, interactive evolutionary reinforcement.

Information retrieval researchers have suggested the use of genetic algorithms to improve the performance of their systems. Gordon (1988), and Blair (1990) have used them to improve document indexing. Chen (1995), Petry et al. (1993), Yang et al. (1992 , 1992), Kraft et al. (1992)

and Sanchez et al. (1992) present an approach based on GAs to enhance the query description. Finally, Gordon (1991) has employed them to build document clusters.

In addition to the drawbacks mentioned above, users must directly get involved in the evolution. That is, these systems need to acquire users' feedback for every generation.

This leads into user's frustration when using the system. On the contrary, with our design, user involvement is needed for providing the feedback only at the initial step of the evolution. In our research we have used the GAs in information retrieval to improve the user profile.

The thesis introduced a number of concepts in the context of information retrieval performance optimisation. User-models were represented by chromosomes in GA, expressed in terms of keywords and weights within the vector space model.

This departs from previous studies, which used binary encoding to indicate the presence or absence of the keyword in the document without any reference to the significance of the keyword to document content. Modification of the representation of user-needs was based on a qualitative relevance metric provided by a fuzzy feedback from the user and a quantitative measure determined from keyword distances (between query and documents). One advantage of this approach is, not only can the system add or remove terms, but can also change the weight of terms. The research also introduced a new genetic algorithm crossover operator to overcome situations where offspring representation can be produced with similar terms, but different weights.

In addition to the drawbacks mentioned above of other research in this area, users must not directly get involved in the evolution. That is, user involvement is needed for providing the feedback only at the initial step of the evolution which leads into inaccurate userprofile. In our system learning mechanism is an automatic and on-line process.

Also another reason that developed adaptive retrieval system performs well is due to the definition of fitness function which has a significant effect upon performance and the effect of user feedback on fitness value. In every user assessment this value is changed according to score values.

8.2. Further Work

A number of issues are identified that could be explored as advancement of this research. There are two interesting directions for future work concerning document indexing. The first, is the issue of indexing documents using concepts, rather than terms. A word can thus be described by a spectrum of related words. By comparing the query words' profiles to generated document index, the system can return articles which are conceptually related to the query words, even if the words themselves do not appear in the text. Since a fuzzy set is defined by enumerating its elements and the degree of membership of each element, it can be used to express word ambiguity by enumerating all possible meanings of a word, then estimating the degrees of compatibilities between the word and the meanings and represent the various meanings of a concept that change dynamically depending on the context. Another approach can be using neural networks in which a node represents a concept and a link represents the strength of the relation between two (connected) concepts.

The second issue is the use of stemming algorithms to extract terms from documents. Stemming is primarily a process of suffix removal which can be done by a set of rules. It will be challenging also, to extend the representation issues to multimedia documents.

The evolutionary algorithm implemented in this research has shown the approach to be promising for modelling adaptive retrieval systems. There are many directions for future work relating to this approach in information retrieval. One area is in automating the modification of genetic operators in response to user actions, for example, by changing crossover and mutation rates.

The user interface design is another interesting area for future research. The amount of user interaction required can be further reduced by use of more intelligent interfaces. A possible addition to our system is to automatically assume positive feedback if the user spends more time on any document. Related to the design of a user interface is the idea of building pro-active behaviour into the system. A web-robot agent could be added to the system to automatically search for articles on the web. Similarly, a scheduler agent could be added to determine multiple searches by different web robots.

REFERENCES

Adeli, H. (1990) *Knowledge Engineering Fundamentals*, Vol. 1, McGraw Hill College.

AgentBuilder [online] (1999) available from: http://www.agentbuilder.com. [Accessed Oct 2003].

Aizawa, A. (2002) "A Co-evolutionary Framework for Clustering in information Retrieval System", *Proceedings of European Symposium on Intelligent Technologies, Hybrid Systems and Implementation on Smart Adaptive Systems (eunite)*, Portugal, 2002.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999) *Modern Information Retrieval*, Addison –Wesley, England.

Baeza-Yates.R and Ribeiro-Neto. N (1999), *Modern Information Retrieval.* Addison Wesley, Essex, England.

Baker, A.D. (1996) "Metaphor or Reality: A Case Study where Agents Bid with Actual Costs to Schedule a Factory", *In:* Clearwatter, S.H. (ed.) *Market-Based Control: A Paradigm for Distributed Resource Allocation*, River Edge, NJ: World Scientific Publishing Co., pp. 184-223.

Balabanovic, M. (1997) "An Adaptive Web Page Recommendation Service", *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, Marina del Ray, CA, 1997.

Balabanovic, M. (1997) *An interface for learning Multi_topic User Profiles from implicit feedback* [online], available from: http://rsv.ricoh.com/~marko/Publications.html. [Accessed Oct 2003].

Balabanovic, M. (1998) "Exploring versus exploiting when learning user models for text recommendation", *User Modeling and User-adapted Interaction*, vol. 8, pp.71-102.

Bargain (1999) *Bargain Bot* [online], available from: http://www.ece.curtin.edu.au/~saunb/bargainbot. [Accessed June 2000].

Beer, M., d'Inverno, M., Jennings, N., Luck, M., Preist, C. and Schroeder, M. (1999) "Negotiation in Multi-Agent Systems", *Knowledge Engineering Review*, vol. 14, no. 3, pp. 285-289.

Belkin, N.J. and Croft, B.W. (1992) "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM*, vol. 35, no. 12, pp. 29-38.

Bellifemine, F. and Poggi, A. (1999) "JADE -A FIPA complaint agent framework", *Proceedings of PAAM'99*, London, April 1999, pp. 97-108.

Bellifemine, F. and Poggi, A. (2000) "An Object Oriented Frame Work to Realize Agent System", *Proceedings of WOA 2000 Workshop*, Parma, May 2000, page 52-57.

Bellifemine, F. and Poggi, A. (2001) *Jade Programmer's Guide* [online], available from: http://sharon.cselt.it/projects/Jade/. [Accessed Oct 2003].

Berenji, H.R. and Vengerov, D.A. (2000) *Learning, Cooperation and Coordination in Multi-Agent System* [online], available from: http://www.iiscorp.com/projects/multi-agent/. [Accessed Oct 2003].

Berney, B. and Ferneley, E. (1999) "CASMIR :Information Retrieval Based on Collaborative User Profiling", *PAAM 1999,* London.

Bigus, J.P. and Bigus, J. (1998) *Constructing Intelligent Agents with Java*, Wiley Computer Publishing.

Boehm, B.W., Brown, J.R. and Lipow, M. (1976) "Quantitative evaluation of software quality", *Proceedings of the IEEE-ACM International Conference on Software Engineering*, 1976, pp. 592-605.

Bojadziev, G. and Bojadziev, M. (1996) *Fuzzy sets, Fuzzy logic, Applications (Advances in fuzzy systems),* vol. 5, World scientific publishing, Singapore.

Bond, A.H. and Gasser, L. (1988) *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, San Mateo.

Booch, G. (1994) *Object Oriented Analysis and Design with Applications*, The Benjamin Company Inc.

Booch, G. and Rumbaugh, J. (1999) *The Unified Modelling Language User Guide*, Addison-Wesley Inc.

Bordogna, G. and Pasi, G. (2002) "Flexible Querying of WEB Documents", *Proceedings of the 17th Symposium on Applied Computing*, Madrid, Spain, 2002, pp. 675-680.

Boughanem, M. and Tmar, M. (2002) "Incremental Adaptive Filtering and Threshold Calibration", *Proceedings of the 17th Symposium on Applied Computing*, Spain, 2002, pp. 640-644.

Bradshaw, J.M. (1997), "An Introduction to Software Agents", *In:* Bradshaw, J.M. (ed.) *Software Agents*, MIT Press, pp. 3-46.

Brenner, W., Zarnekow, R. and Witting, H. (1998) *Intelligent Software Agent*, Springer.

British Telecommunications Labs (1999) *ZEUS* [online], available from: http://www.labs.bt.com/projects/agents/zeus/index.htm. [Accessed Oct 2003].

Broker agent collections [online] (1999) available from: http://www.swi.psy.uva.nl/projects/IBROW3/related-brokers.html. [Accessed Oct 2003].

Brooks, R.A. (1986) "A Robust Layered Control System for a Mobile Robot", *IEEE Journal on Robotics and Automation*, vol. 2, pp.14-23.

Brusilovsky, P., and Schwarz, E. "User as Student: Towards an Adaptive Interface for Advanced Web-Based Applications." In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), *User Modeling: Proceedings of the Sixth International Conference, UM97*. Vienna, New York: Springer Wien New York, 1997, pp. 177-188.

Brusilovsky, P., InterBook Home Page [online]. Pittsburgh, PA, April 1999. Available from the World Wide Web: http://www.contrib.andrew.cmu.edu/~plb/InterBook.html [Accessed March 2004].

Bruza, P.D. and Song, D. (2002) "Inferring Query Models by Computing Information Flow", *Proceedings of the 11th International Conference on Information and Knowledge Management*, Virginia, 2002, pp. 260-269.

Buckley, C. and Voorhees, E.M. (2000) "Evaluating Evaluation Measure Stability", *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* ACM press, 2000, pp.33-40.

Bussmann, S., Jenning, N.R. and Wooldridge, M. (2000) On the Identification of Agent in the Design of Production Control System, *Proceedings of Agent-Oriented Software Engineering (AOSE2000)*, Ireland, June 10, 2000, Lecture Notes in Computer Science, no. 1957, Springer-Verlag, Germany.

Callan, J. (1997) *Learning while filtering document* [online], available from: www.cs.umass.edu/~callan/. [Accessed Oct 2003].

Cernuzzi, L. and Giret, A. (2000) "Methodological Aspects in the Design of a Multi-Agent System", *Proceedings of AOIS 2000*, Stockolm, Sweden, June 5-6, 2000.

Chau, R. and Yeh, Ch. (2000) "Explorative Multilingual Text Retrieval Based on Fuzzy Multilingual Keyword Classification", *Proceedings of the Fifth International Workshop on Information Retrieval with Asian Language*, China, 2000, pp 33-40.

Chen, H. and Dhar, V. (1995) "Cognitive process as a basis for intelligent retrieval systems", *Information Processing & Management*, vol. 27, pp. 405-432.

Chen, H. and Shankaranarayanan, G. (1998) "A Machine Learning Approach to Inductive Query by Examples", *Journal of American Society For Information Science,* vol 49, no. 8, pp. 693-705.

Chen. P and Kuo. F (2000) "An information retrieval system based on a user-profile", *The Journal of Systems and Software*, 54(1), pp.3-8.

Cox, E. (1998) *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, & Maintaining Fuzzy Systems*, 2nd edition, AP Professional, USA.

Crestani, F. (1995) "Implementation and Evaluation of a Relevance Feedback Device Based on Neural Networks", *In:* Mira, J. and Cabestany, J. (eds.) *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks*, Malaga, Spain, June 1995, Lecture Notes in Computer Science, no. 930, Springer-Verlag, pp. 597–604.

Croft, W.B. (1993) "Knowledge-based and statistical approaches to text retrieval", *IEEE Expert*, vol. 8, no. 2, pp. 8–12.

Decker, K. and Lesser, V. (1995) "Macron: An architecture for multi-agent cooperative information gathering", *In: CIKM Conference, Workshop on Intelligent Information Agents.*

Doran, J.E., Franklin, S., Jennings, N.R. and Norman, T.J. (1997) "On Cooperation in Multi-agent Systems", *The Knowledge Engineering Review*, vol. 12, no. 3.

Edgar, E. (1998) *Profusion Personal Assistant: An Agent for Personalized Information Filtering on WWW*, MSc Theses, University of Kansas.

Eiter, T. and Mascardi, V. (2002) "Comparing Environments for Developing Software Agents", *The European Journal on Artificial Intelligence (AI Communications),* vol. 15. no. 4, pp. 169-197.

Encarnação, L. Miguel, "Multi-level user support through adaptive hypermedia: a highly application-independent help component." *Proceedings of the 1997 international conference on Intelligent user interfaces*, 1997, pp. 187-194.

Falasconi, S., Lanzola, G. and Stefanelli, (1997) "An Ontology-Based Multi-Agent Architecture For Distributed Health-Care Information Systems", *Methods Of Information In Medicine,* pp. 20-29.

Familiar, M. (1997) "Machine Learning and Human-agent Interaction", *Workshop HAI-97* [online], available from: http://www.ipr.ira.uka.de/~kaiser/events/hai97. [Accessed Dec 1999].

Fan, Y. and Gauch, S. (1997) *An Adaptive Multi-Agent Architecture for the ProFusion Meta Search System* [online], available from: http://www.tisl.ukans.edu/~sgauch/papers/webnet97.html. [Accessed Oct 2003].

Fini, T., Labrou, Y. and Mayfield, J. (1997) "KQML as an Agent Communication Language", *In:* Bradshaw, J.M. (ed.) *Software Agents*, MIT Press, pp. 291-316.

Fink, J., Kobsa, A., and Nill, A. "User-oriented Adaptivity and Adaptability in the AVANTI Project." Conference *Designing for the Web: Empirical Studies*. Microsoft Usability Group, Redmond, WA, 1996. [Accessed March 2003]:

Fink, Joseph, Alfred Kobsa, and Andreas Nill, "Adaptable and Adaptive Information Access for All Users, Including the Disabled and the Elderly." In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), *User Modeling: Proceedings of the Sixth International Conference, UM97*. Vienna, New York: Springer Wien New York, 1997, pp. 171-173.

Finlay, J. and Dix, A. (1996) *An Introduction to Artificial Intelligence.* UCL Press, Taylor and Francis.

FIPA: Foundation for Intelligent Physical Agents (1997) *FIPA97, Part 2, versions 1.0 and 2.0: Agent Communication Language Specification* [online], available from: http://www.fipa.org/repository/fipa97.php3. [Accessed Oct 2003].

Frakes, W.B. and Baeza-Yates, R. (1992) *Information Retrieval Data structure and Algorithms,* Prentice Hall.

Franklin. S, Graesser. A. (1996) "Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents", Intelligent Agents III: Agent Theories, Architectures, and Languages, Proceedings *of ECAI'96 Workshop(*ATAL), Hungary, Aug.

Galan, A. (1997) *JAFMAS, A Java-based Agent Framework for Multi-Agent Systems Development and Implementation* [online], Technical Report, University of Cincinnati, ECECS Department, available from: http://www.ececs.uc.edu/~abaker/JiVE. [Accessed Oct 2003].

Gary. B. Shelly, Thomas J.Cashman, Harry J.Rosenblatt, *System analysis and design,* 5th addition, Thomson Course Technology, USA.

Gauch, S. and Wang, G. (1996) "ProFusion: Intelligent Fusion from Multiple Distributed Search Engines", *Proceedings of WebNet96: The First World Conference of the Web Society*, San Francisco, CA, October 1996.

Genesereth, M. (1997) "An Agent-based Framework for Interoperability", *In:* Bradshaw, J.M. (ed.) *Software Agents*, MIT Press, pp. 317-345.

Gervais, M.P. (2002) ODAC : "An Agent-Oriented Methodology Based on ODP" [online], *Journal of Autonomous Agents and Multi-Agent Systems*, Jan 2002, available from: http://www-src.lip6.fr/homepages/Marie-Pierre.Gervais/AGNT74-00.pdf. [Accessed Oct 2003].

Ginsberg, M.L. (1991) "Knowledge Interchange Format: The KIF of Death", *AI Magazine*, Fall 1991, pp. 57-63.

Giunchiglia, F., Mylopoulos, J. and Perini, A. (2002) "The Tropos Software Development Methodology: Processes, Models and Diagrams", Submitted to: *Autonomous Agent-MAS '02, A Knowledge Level Software Engineering*, Technical Report, no. 0111-20, ITC - IRST, Nov 2001.

Goldberg, D.E. (1989) *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley.

Graham, I., Henderson-Seller, B. and Younessi, H. (1997) *The OPEN process specification*, Addison-Wesley.

Greiff, W.T., Morgan, W.T. and Ponte, M.T. (2002) "The role of variance in term weighting for probabilistic information retrieval", *Proceedings of the 11th International Conference on Information and Knowledge Management*, McLean, Virginia, pp. 252-259.

Gruber, T.R. (1993) "A Translation Approach to Portable Ontology Specifications", *Proceedings of the Knowledge Acquisition for Knowledge-Based Systems (KAW'93)*, Canada, pp. 199-220.

Gudivada, V.N., Raghavan, V.V., Grosky, W.I. and Kasanagottu, R. (1997) "Information Retrieval on the World Wide Web", *IEEE Internet Computing*, September-October 1997.

Hare, G. and Jenning, N.R (1996) *Foundations of Distributed Artificial Intelligence*, John Wiley and sons.

Harries, S. (1993) *Networking and telecommunications for information systems: An introduction to information networking*, Library Association, 1993.

Haudeneder, H. and Striner, D. (1996) *Multi-agent Cooperation-Concept and Applications*, Lecture Notes in Computer Science, no. 1128, Springer-Verlag, pp. 195-197.

Hayes-Roth, B. (1995) "An Architecture for Adaptive Intelligence System", *Artificial Intelligence*, vol. 72, pp. 329-365.

Hayes-Roth, B., Uckun, S., Larsson, J.E., Gaba, D., Barr, J. and Chien, J. (1994) "Guardian: A prototype intelligent agent for intensive-care monitoring", *Proceedings of the National Conference on Artificial Intelligence*, pp. 1503-1511.

Hideyuki, T. (2001) "Interactive Evolutionary Computation as Humanized Computational Intelligence Technology", *International Conference on Computational Intelligence, Theory and Applications: 7th Fuzzy Days*, Dortmund, Germany, October 1-3, 2001.

Holland, J. (1992) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology,* MIT Press.

Horng, J.T. and Yeh, C.C. (2000) "Applying Genetic Algorithms to Query Optimization in Document Retrieval", *Information Processing and Management,* vol. 36, no. 5, pp.737-759.

Horvitz, Eric, Jack Breese, David Heckerman, David Hovel, and Koos Romelse. "The Lumière Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users." *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence,* Madison, WI, July 1998. Morgan Kaufmann: San Francisco, 1998, pp. 256-265.

http://fit.gmd.de/hci/projects/avanti/publications/ms96.html.

Hull, D. (1993) "Using Statistical Testing in the Evaluation of Retrieval Performance", *Proceedings of the 16th ACM SIGIR Conference*, pp. 329-338.

Imam, F. and Kodratoff, Y. (1997) "Intelligent Adaptive Agents", *Proceedings of AAAI-96 workshop on IAA*, pp. 75-80.

Jansen, B.J., Spink, A., Bateman, J. and Saracevic, T. (1998) "Real life information retrieval: a study of user queries on the Web", *ACM SIGIR Forum*, vol. 32, no. 1, April, pp. 5-17.

Jarvelin, K. and Kekalainen, J. (2000) "IR evaluation methods for highly relevant documents", *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM press, 2000, pp. 41-48.

Jayaratna, N. (1994) *Understand and Evaluation Methodologies, NIMSAD, A Systemic Framework*, McGraw Hill, London.

Jenning, N.R. and Wooldrige, M. (1998) *Agent Technology Foundation, Application and Markets*, Springer.

Jenning, N.R., Sycara, K. and Wooldrige, M. (1998) "A Roadmap of Agent Research and Development", *The Journal of Autonomous Agents and Multi-Agent Systems*, vol. 1, pp. 7-38.

Keen, M.E. (1997) "Presenting Results of Experimental Retrieval Comparisons", *In:* Sparck-Jones, K. and Willett, P. (eds.) *Readings in Information Retrieval*, Morgan Kaufmann Publishers. pp. 217-222.

Kinny, D., Georgeff, M. and Rao, A. (1996) "A methodology and modelling technique for systems of BDI agents", *In:* Van der Velde, W. and Perram, J. (eds.) *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, Lecture Notes on Artificial Intelligence, no. 1038, Springer-Verlag: Heidelberg, Germany.

Kosba, A. (2001) "Generic User Modeling Systems", *User Modelling and User- Adapted Interaction,* vol 11, pp. 49-63, 2001.

Kosko, B. (1994) "Fuzzy Systems as Universal Approximators", *IEEE Transactions on Computers*, vol. 43, no. 11, pp. 1329-1333, November 1994.

Koza, J.R. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT press.

Kwok, K.L. (1995) "A Network Approach to Probabilistic Information Retrieval", *ACM Transactions on Information Systems,* vol. 13, no. 3, pp. 324-353.

Lau, R. and Hofestede, A.H.M. (2001) "Nonmonotonic reasoning for adaptive information filtering", *Proceedings of 24th Australian conference on computer science*, Australia, 2001, pp. 109-116.

Laukkanen, M. (2000) *Evaluation of JADE 1.2* [online], available from: http://sharon.cselt.it/projects/Jade/. [Accessed: 16 Feb. 2000].

Lawrence, S. and Giles, C.L. (1998) "Context and Page Analysis for Improved Web Search", *IEEE Internet Computing,* July 1998, pp. 38-46.

Lee, C.C. (1990) "Fuzzy Logic in Control Systems", *IEEE Transactions, SMC*, pp. 404-435.

Lee, J.H. (1994) "Properties of Extended Boolean Models in Information Retrieval", *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July 3-6, 1994, pp. 182-190.

Lesser, V.R. and Corkill, D.D. (1987) "Distributed problem solving", *In: Encyclopedia of Artificial Intelligence*, John Wiley & Sons, pp. 245-251.

Lewis, D. (1992) "An evaluation of phrasal and clustered representations on a text categorization task", *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Denmark, 1992, pp. 37-50.

Light, M. and Maybury, M.T. (2002) "Personalized Multimedia Information Access", *Communication of the ACM*, Special Issue: The Adaptive Web, vol. 45, no. 5, pp. 54-59.

Ljungberg, M. and Lucas, A. (1992) *The OASIS: Air Traffic Management System*, Technical Note 28, August 1992.

Losee, R.M. (1998) "Comparing Boolean and Probabilistic Information Retrieval Systems Across Queries and Disciplines", *Journal of American Society for Information Science*, vol. 48, no. 2, pp. 143-156.

M. D. Gordon, "User-based document clustering by redescribing subject descriptions with a genetic algorithm", *Journal of the American Society for Information Science*, 42(5):311-322, June 1991.

M. Gordon, "Probabilistic and genetic algorithms for document retrieval", *Communications of the ACM*, 31(10):1208-1218, October 1988.

Maclin, R. and Shavlik, J. (1996) "Creating Advice-taking Reinforcement Learners", *Machine Learning*, vol. 22, no. 1-3, pp.123-137.

Maes, P. (1995) "Artificial Life Meets Entertainment: Life like Autonomous Agents", *Communications of the ACM,* vol. 38, no. 11, pp. 108-114.

Maglio, P.P. and Barrett, R. (1997) "How to Build Modelling Agents to Support Web Searchers", *Proceedings of User Modelling (UM'97)* [online], available from: http://um.org. [Accessed Oct 2003].

Mcrobb, S. and Farmer, R. (1999) *Object-Oriented Systems Analysis and Design,* Mc Graw Hill.

Mea, V. and Mizzaro, S. (2004) "Measuring retrieval effectiveness: A new proposal and first experimental validation", *Journal of American Society For Information Science and Technology,* vol 55, issue. 6, pp. 530-543.

Meadow, C.T. (1992) *Text information retrieval systems.* Academic Press, Inc., San Diego, California.

Menczer, F. and Belew, R.K. (1994) Evolving sensors in environments of controlled complexity. *In:* Brooks, R. and Maes, P. (eds.) *Artificial Life IV,* MIT Press.Cambridge, MA,

MESSAGE: Methodology for Engineering Systems of Software Agents (2001) *Methodology for Agent-Oriented Software Engineering* [online], September 2001, available from: http://www.eurescom.de/public/projectresults/P900-series/907ti1.asp. [Accessed Oct 2003].

Michalewicz, Z. (1996) *Genetic Algorithms + Data structures = Evolution Programs,* 3rd rev. and extended ed., Springer-Verlag.

Minio, M. and Tasso, C. (1996) "User Modeling for Information Filtering on INTERNET Services: Exploiting an Extended Version of the UMT Shell", *Proceedings of the 5th International Conference on User Modelling,* Hawaii, January 1996.

Mooers, C.N. (1950) "Putting Probability to Work in Coding Punched Cards: Zatocoding", *Zator Technical Bulletin,* no. 10, 1947, Reprinted as *Zator Technical*

Bulletin, no. 30, 1950, *cited in*: Calvin N Mooers papers [online], available from: http://www.cbi.umn.edu/collections/inv/mooers.htm. [Accessed Oct 2003].

Mooers, C.N. (1976) "Technology of Information Handling: A pioneer's View", *Bulletin of the American Society for Information Science*, vol. 2, no. 8, pp. 18-19.

Moukas, A. and Maes, P. (1998) "Amalthaea: an evolving multi-agent information filtering and discovery system for the WWW", *Autonomous Agents and Multi-agent Systems*, vol. 1, no. 1, pp. 59-88.

Mullender, S. (1993) *Distributed System*, ACM Press, Addison Wesley.

Nahm and Mooney (2001) *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, Seattle, WA, August 2001, pp. 979-984.

Newell, S.C. (1997) "User Models and Filtering Agents for Improved Internet Information Retrieval", *User Modeling and User-adapted Interaction*, vol. 7, pp. 223- 237.

Odell, J., Parunak, H.V.D. and Bauer, B. (2000) "Extending UML for Agents", *In:* Wagner, W., Lesperance, Y. and Yu, E. (eds.) *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence, AAAI'00.*

Othman, Z.A. (2003) "*The use of Software Abstraction to Develop Agent System*", Sheffield Hallam University, Unpublished Phd Thesis.

Padgham, L. and Winikoff, M. (2002), "Prometheus: A Methodology for Developing Intelligent Agents", *Proceedings of the 3rd International Workshop on AgentOriented Software Engineering (AAMAS'02)*, Bologna, Italy, July 2002.

Paice, C.P. (1984) "Soft Evaluation of Boolean Search Queries in Information Retrieval Systems", *Information Technology: Research and Development*, vol. 3, no. 1, pp. 33-42.

Pazzani, M.J. (2000) "Representation of Electronic Mail Filtering: A User Case Study", *Proceedings of the 5th International Conference on Intelligent User Interfaces*, USA, 2000, pp. 202-206.

Porter, M. (1997) "An algorithm for suffix stripping", *In:* Sparck-Jones, K. and Willett, P. (eds.) *Readings in Information Retrieval*, Morgan Kaufmann, pp. 313-316.

Possas, B., Meria, W. and Zivani, N. (2002) "Set-based model: New Approach for Information Retrieval", *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Finland, pp. 230-237.

Poulter, A (1997) "The Design of World Wide Web Search Engines: A Critical Review", *Program*, vol. 31, no. 2, April 1997, pp. 131-145.

Ranky, P.G. (1994) *Concurrent/Simultaneous Engineering: Methods, Tools and Case Studies*, CIMware.

Reichert, M. and Dadam, P. (1998) "ADEPT flex Supporting Dynamic Changes of Workflows Without Loosing Control", *Journal of Intelligent Information System.*

Reid, J. (2000) "A Task-oriented Non-interactive Evaluation Methodology for Information Retrieval Systems", *Information Retrieval*, vol. 2, pp. 115-129.

Ricordel, P.M. and Demazeau, Y. (2000) "From analysis to deployment: a multiagent platform survey", *Proceedings of 1st International Workshop on Enginnering Societis in the Agents World (ESAW), ECAI'2000*, November 2000, Berlin, Germany, Springer Verlag, pp. 93-105.

Roberston, S.E. (1997) "The Probability Ranking Principle in IR", *In:* Sparck-Jones, K. and Willett, P. (eds.) *Readings in Information Retrieval*, Morgan Kaufmann, pp. 281-286.

Rumbaugh., J. (1991) *Object-Oriented Modelling and Design*, Prentice- Hall.

Russell, J.S and Norvig, P. (1995) *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice Hall.

S. Kraus, S. (1997) "Negotiation and Cooperation in Multi-agent Environment", *Artificial Intelligence*, vol. 94, pp. 79-97.

Salton G. and McGill, M. (1983) *Introduction to Modern Information Retrieval*, Mc GrawHill, New York.

Salton, G. and McGill, M.J. (1997) "The SMART and SIRE Experimental Retrieval System", *In:* Sparck-Jones, K. and Willett, P. (eds.) *Readings in Information Retrieval*, Morgan Kaufmann, pp. 381- 399.

Salton, G. and Buckley, B. (1998) "Term Weighting Approaches in Automatic Text Retrieval", *Information Processing and Management*, vol. 24, no.5 pp. 513-523.

Salton, G. and Buckley, C. (1997a) "Term Weighting Approaches in Automatic Text Retrieval", *In:* Sparck-Jones, K. and Willett, P. (eds.) *Readings in Information Retrieval*, Morgan Kaufmann, pp. 322-328.

Salton, G. and Buckley, C. (1997b), "Improving Retrieval Performance by Relevance Feedback", *In:* Sparck-Jones, K. and Willett, P. (eds.) *Readings in Information Retrieval*, Morgan Kaufmann, pp. 355-363.

Salton, G. and Buckley, C. (1998) "Term Weighting Approaches in Automatic Text Retrieval", *Information Processing and Management*, vol. 24, no.5, pp. 513-523.

Salton, G. and Wong, A. (1997) "A Vector Space Model for Automatic Indexing", *In:* Sparck-Jones, K. and Willett, P. (eds.) *Readings in Information Retrieval*, Morgan Kaufmann, pp. 273-280.

Saracevic, T. (1995) "Evaluation Of Evaluation In Information Retrieval", *In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* Seattle, Washington, USA, July 9-13, pp. 138-146.

Sarma, V. (1996) "Intelligent Agents", *Journal of IETE*, vol. 42, no. 3, pp. 105-109.

Schroeder, M. and Mora, I. (1996) "A Deliberative and Reactive Diagnosis Agent Based on Logic Programming", *Proceedings of Workshop on Agent Theory and Language 96*, pp. 248-307.

Sebastiani, F. (2002) "Machine Learning in automated text categorization", *ACM Computing Surveys (CSUR),* vol. 34, no. 1, pp. 1-47.

Sen, S. (1996) "Adaptation, Coevolution and Learning in Multiagent Systems", *AAAI 1996, Spring Symposium*, Technical Report, SS-96-0.

Sen, S. (ed.) (1995) *Working Notes of the IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, Montral, Canada.

Seo, Y. and Zhang, B. (2000) "A Reinforcement Learning Agent for Personalized Information Filtering", *Proceedings of the 5th International Conference on Intelligent User Interfaces*, USA, pp. 248-251.

Smith, D.C., Cypher, A. and Spohrer, J. (1994) "KidSim: Programming Agents Without a Programming Language", *Communications of the ACM,* vol. 37, no. 7, pp. 55-67.

Smith, J.M. (1989) *Evolutionary Genetics*, Oxford University press, New York.

Smith, R.E. (2002) *BEAGLE: A Genetic Algorithm for Information Filter Profile Creation* [online], available from: http://www.cis.uab.edu/info/alumni/sf/Papers. [Accessed Oct 2003].

Soderland, S. (1997) "Learning to Extract Text-base Information from the World Wide Web", *Proceedings of 3rd International Conference on Knowledge Discovery and Data Mining.*

Sparck-Jones, K. (1974) "Understanding Natural Language", *International Journal of Man-Machine Studies*, vol. 6, no. 2, pp. 277-284.

Sparck-Jones, K. (1981) *Information Retrieval Experiment.* London: Butterworths.

Spark-Jones K. and Willet, P. (eds.) (1997) *Readings in Information Retrieval.* Morgan Kaufmann, San Francisco, California.

Srinivasan, P., Ruiz, M.E., Kraft, D.H. and Chen, J. (2001) "Vocabulary mining for information retrieval: rough sets and fuzzy sets", *Information Processing and Management*, vol. 37, no. 1, pp. 15-38.

Sutton, S. and Barto, G. (1998) *Reinforcement Learning*, MIT press.

Tools (2001) Agent Framework Development Tools [online] (2001) available from: http://www.agentbuilder.com/agentTools/index.html. [Accessed Oct 2003].

TREC (2000) Text Retrieval Conference [online], available from: http://trec.nist.gov/presentations/presentations.html. [Accessed Oct 2003].

Treur, J., Jonker, C.M. and Brazier, F.M.T. (1999) *Design of Intelligent Multi-agent System*, Course Notes, 10-16 February 1999, Computer Science Department, Vrije University, Amsterdam.

UM 97 Reader's Guide." *User Modeling: Proceedings of the Sixth International Conference, UM97.* On-line proceedings, 1997. http://www.um.org [Accessed March 2004]

Van Rijsbergen, C.J. (1979) *Information Retrieval*, 2nd edition, Butterworths, London.

Vrajitoru, D. (1998) "Crossover Improvement for the Genetic Algorithm in Information Retrieval", *Information Processing & Management*, vol. 34, no. 4, pp. 405-415.

Wavish, P. and Graham, M. (1996) "A Situated Action Approach to Implementing Characters in Computer Games", *International Journal of Applied Artificial Intelligent*, vol. 10, no. 1, pp. 53-74.

Weiss, G. (1999) *Multiagent System, A Modern Approach to DAI*, The MIT Press, London, UK.

Wenger, D. and Probst, A.R. (1998) "Adding Value with Intelligent Agents in Financial Services", *In:* Jenning, N.R. and Wooldridge, M.J. *Agent Technology Foundations, Applications and Markets,* Springer Verlag, Germany.

Wirfs-Brock, R. and Wilkerson, B. (1990) *Designing Object-Oriented Software*, Pressman.

Witten, I.H., Moffat, A. and Bell, C. (2000) *Managing Gigabytes*, Morgan Kafmann.

Wondergem, B.C.M., Bommel, P.V., Huibers, T.W.C., Weide, T.V. (1997) "Towards an Agent-Based Retrieval Engine", *In:* Funer, J. and Harpers, D.J. *Proceedings of the 19th BCS-IRSG Colloquium*, Aberdeen, Scotland, pp. 124-144.

Wooldridge, M., Jennings, N.R. and Kinny, D. (2000) "The Gaia Methodology for Agent-Oriented Analysis and Design", *Autonomous Agents and Multi-Agent Systems*, vol. 3, no. 3, pp. 285-312.

Wooldrige, M. and Jenning, N.R. (1995) "Intelligent Agent: Theory and practice", *Knowledge Engineering Review*, vol. 10, no. 2.

Wu, M.M. and Sonnenwald, D.H. (1999) *Reflections on Information Retrieval Evaluation* [online], National Taiwan Normal University/ University of California at Chapel Hill, Taiwan/USA, available from: http://pnclink.org/annual/annual1999/1999pdf/wu-mm.pdf. [Accessed Oct 2003].

Yang, C. and Yen, J. (2000) "Intelligent internet searching agent based on hybrid simulated annealing ", *Decision Support System,* vol. 28, pp. 269- 277.

Zadeh, L.A. (1965) "Fuzzy Sets", *Information and Control*, no. 8, pp. 338-53.

Zadeh, L.A. (1984) "Making Computers Think Like People", *IEEE Spectrum*, August 1984.

APPENDIX

AgentBuilder [AgentBuilder 1999]

AgentBuilder is an integrated tool suite for constructing intelligent software agents. It consists of two major components, the Toolkit and the Run-Time System. The Agent-Builder Toolkit includes tools for managing the agent-based software development process, analysing the domain of agent operations, designing and developing networks of communicating agents, defining behaviours of individual agents, and debugging and testing agent software. Agents communicate using KQML.

JAFMAS [Galan 1997]

The Java Agent Framework for Multi-Agent Systems (JAFMAS) provides a generic methodology for developing speech-act-based multi-agent systems, an agent architecture and a set of classes to support the implementation of these agents in Java. JAFMAS provides communication, linguistic and coordination support. Communication support is provided for both directed communication and subject-based broadcast communication. Linguistic support is provided through a speech-act (KQML) based communication language, which provides an agent-independent semantics. Agent plans and their coordination are conceptualised as rule-based conversations represented by automata models.

JADE [Bellifemine and Poggi 1999]

JADE (Java Agent Development Environment) is a software framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middle-ware that claims to comply with the FIPA specifications (see section 6.2.2) and, a set of tools that supports the debugging and deployment phase. The agent platform can be distributed across machines (which do not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can even be changed at run-time by moving agents from one machine to another one, as and when required. The communication architecture offers flexible and efficient messaging, where JADE creates and manages a queue of incoming ACL messages, private to each agent; agents can access their queue via a combination of several modes:

blocking, polling, timeout and pattern matching. The full FIPA communication model has been implemented and its components have been clearly distinguished and fully integrated: interaction protocols, envelope, ACL, content languages, encoding schemes, ontologies and, transport protocols. JADE has also been integrated with JESS, a Java shell of CLIPS, in order to exploit its reasoning capabilities [Bellifemine and Poggi 2001, Bellifemine 2001].

JATLite [Stanford University 1999]

The Java Agent Template Lite (JATLite) is a package of programs written in the Java language that allow users to create new software agents that communicate over the Internet. JATLite provides a basic infrastructure in which agents register with an Agent Message Router facilitator using a name and password, connect/disconnect from the Internet, send and receive messages, transfer files, and invoke other programs or actions on the various computers where they are running. JATLite provides a template for building agents that utilise a common high-level language and protocol so that it becomes easy to build systems in a common way, but without imposing any particular theory of autonomous agents. Agent communication is based on KQML messages.

ZEUS [British Telecommunications Labs 1999]

ZEUS is a tool-kit for the development of collaborative agents, written in Java. Each ZEUS agent consists of a definition layer, an organisational layer and a coordination layer. The Definition Layer comprises the agent's reasoning and learning abilities, its goals, resources, skills, beliefs, and preferences. The Organisation Layer describes the agent's relationships with other agents, for example, what agencies it belongs to, what abilities it knows other agents possess, etc. At the Coordination Layer the agent is modeled as a social entity, in terms of the coordination and negotiation techniques it possesses. Built on top of the coordination layer are the communication protocols that implement inter-agent communication. Beneath the definition layer is the API that links the agent to the physical realisations of its resources and skills.