Department of Electronics, Computer Science and Automatic Control

PhD Thesis

Collaborative Recommender Agents Based On Case-Based Reasoning and Trust

Thesis presented by Miquel Montaner,
to obtain the degree of:
PhD in Computer Engineering.

Supervisors:
Dr. Josep Lluís de la Rosa
Dr. Beatriz López

Girona, September 2003

Collaborative Recommender Agents Based On Case-Based Reasoning and Trust

by

Miquel Montaner

To my family.
A la meva família.

Acknowledgments

This work was supported by the UdG Research Grant BR99/762, the Spanish CICYT Project TAP98-0955-C03-02, the Spanish MCYT Project DPI2001-2094-C03-01 and the Agentcities.NET Project ACNET.02.50.

I would also like to thank certain people for their inestimable support during these four short years since, without their help, I would not have been able to finish my work.

First of all, I want to mention my first supervisor, Josep Lluís de la Rosa. He is the person who most encouraged me while working at the Agents Research Laboratory. I also want to thank him for all the resource material he provided for my work.

My most special gratitude to Beatriz López, my second but no less important supervisor. She has guided my steps through these years and I have learnt so much from our interminable discussions. I want to thank her collaboration because I could not have achieve it without her leadership.

I also wish to express my sincere appreciation to the people at Intelligent Software Agents Group at the Robotics Institute of the Carnegie Mellon University where part of this work was developed during a research stage. Special gratitude to Professor Katia Sycara, head of the group, who accepted me as research visitor.

Special thanks to my XXX PhD-mates (Xavier Armangué, Xavier Muñoz and Xavier Lladó). Although we belong to different research groups, I greatly appreciate their advice and especially, their patience.

Finally, I would like to dedicate this achievement to my family, my girlfriend and my friends. They have always believed in me, a priceless motivation which I truly appreciate. I hope that I have not disappointed them.

Abstract of the Thesis

The Artificial Intelligence (AI) community has carried out a great deal of work on how AI can help people to find out what they want on the Internet. As a result, the idea of recommender systems has been widely accepted among users. The main task of a recommender system is to locate items, information sources and people related to the interest and preferences of a single person or a group of people. This involves the construction of user models and the ability to anticipate and predict user preferences.

This thesis focusses on the study of AI techniques which improve the performance of recommender systems. Initially, a detailed analysis of the current state-of-the-art in this field has been carried out. This work has been organised as a taxonomy where existing recommender systems on the Internet are classified into 8 general dimensions. This taxonomy provides us with an indispensable knowledge base from which to design our proposal.

Secondly, this thesis proposes a new CBR approach to recommendation. Case-based reasoning (CBR) is a paradigm for learning and reasoning through experience suitable for recommender systems due to its being based on human reasoning. We provide a forgetting mechanism for case-based profiles that controls the relevance and age of past experiences. Experimental results show that this proposal better adapts the profiles to users and solves the utility problem of CBR systems.

Thirdly, this thesis proposes the "agentification" of recommender systems in order to take advantage of interesting agent properties such as proactivity, encapsulation or social ability. Recommender systems sharply improve the quality of results when information about other users is utilised when recommending a given user. Collaboration among agents is performed with the opinion-based filtering method and the collaborative filtering method through trust. Both are based on a social model of trust making agents less vulnerable to others while collaborating. Experimental results show that our collaborative recommender agents improve the performance of the system while preserving the privacy of the user's personal data.

Finally, this thesis also proposes an evaluation procedure for recommender systems that allows a scientific discussion of the results. This proposal simulates the users behaviour over time based on real user profiles. We hope this new evaluation methodology will contribute towards the progress in this area of research.

Resum de la Tesi

La comunitat científica que treballa en Intel·ligència Artificial (IA) ha dut a terme una gran quantitat de treball en com la IA pot ajudar a les persones a trobar el que volen dins d'Internet. La idea dels sistemes recomanadors ha estat extensament acceptada pels usuaris. La tasca principal d'un sistema recomanador és localitzar ítems, fonts d'informació i persones relacionades amb els interessos i preferències d'una persona o d'un grup de persones. Això comporta la construcció de models d'usuari i l'habilitat d'anticipar i predir les preferències de l'usuari.

Aquesta tesi està focalitzada en l'estudi de tècniques d'IA que millorin el rendiment dels sistemes recomanadors. Inicialment, s'ha dut a terme un anàlisis detallat de l'actual estat de l'art en aquest camp. Aquest treball ha estat organitzat en forma de taxonomia on els sistemes recomanadors existents a Internet es classifiquen en 8 dimensions generals. Aquesta taxonomia ens aporta una base de coneixement indispensable pel disseny de la nostra proposta.

En segon lloc, aquesta tesi planteja una nova proposta de CBR aplicat al camp de la recomanació. El raonament basat en casos (CBR) és un paradigma per aprendre i raonar a partir de la experiència adequat per sistemes recomanadors degut als seus fonaments en el raonament humà. Proposem un mecanisme d'oblit per perfils basats en casos que controla la rellevància i edat de les experiències passades. Els resultats experimentals demostren que aquesta proposta adapta millor els perfils als usuaris i soluciona el problema de la utilitat que pateixen el sistemes basats en CBR.

En tercer lloc, aquesta tesi proposa l'agentificació dels sistemes recomanadors per tal de treure profit de propietats interessants dels agents com ara la proactivitat, la encapsulació o l'habilitat social. Els sistemes recomanadors milloren espectacularment la qualitat dels resultats quan s'utilitza informació sobre els altres usuaris al recomanar a un usuari concret. La col·laboració entre agents es realitza a partir del mètode de filtratge basat en la opinió i del mètode col·laboratiu de filtratge a partir de confiança. Els dos mètodes es basen en un model social de confiança que fa que els agents siguin menys vulnerables als altres quan col·laboren. Els resultats demostren que els agents recomanadors col·laboratius proposats milloren el rendiment del sistema mentre que preserven la privacitat de les dades personals de l'usuari.

Finalment, aquesta tesi també proposa un procediment per avaluar sistemes recomanadors que permet la discussió científica dels resultats. Aquest procediment simula el comportament dels usuaris al llarg del temps a partir de perfils d'usuari reals. Esperem que aquesta nova metodologia d'avaluació contribueixi al progrés en aquesta àrea de recerca.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter describes the motivation leading to the presentation of this thesis. The objectives of this thesis and the subjects included in this document are briefly explained. The chapter ends describing the structure and contents of this thesis.

1.1 Motivation

The development of the Internet has resulted in a global information society with a growing number of users around the world. Yet, because of this avalanche of information at our door step, there is a rapidly increasing difficulty in finding what we want when we need it and in a manner which best meets our requirements. Users are constantly confronted with situations in which they have too many options to choose from, where they need help to explore and filter out their preferences from the myriad possibilities. Internet Search Engines, designed originally to be helpful, now commonly find many thousands of potentially relevant sites, thus losing their usefulness.

Recently, in the Artificial Intelligence (AI) community, there has been a great deal of work on how AI can help to solve this problem. The idea of *recommender systems* [Resnick 97] has been widely accepted among users who require assistance in searching, sorting, classifying, filtering and sharing the vast amount of information now available on the Web. The main task of a recommender system is to locate items, information sources and people related to the interest and preferences of a single person or a group of people. This involves the construction of user models

and the ability to anticipate and predict user preferences.

In addressing these types of tasks, recommender systems draw on previous results from machine learning and other AI technology advances. Among the various machine-learning technologies, our research group has been traditionally concentrated on *Case-Based Reasoning* (CBR) [Aamodt 94] as a paradigm for learning and reasoning through past experiences as humans do. Particularly, we have successfully applied CBR to process supervision and automatic control because of its maintainability, explicability, robustness and exception handling features (i.e., [Meléndez 01, Pous 03, Macaya 02, Meléndez 03]), which make CBR perform better than any other knowledge representation scheme. From such experience on the system engineering domain, we think on taking advantage of these CBR properties in other domains such as Internet, in which the interest of the research group has recently shifted. Then, our purpose is to design a recommender system based on the CBR methodology. However, when we apply CBR to recommender systems, several drawbacks arise. Among them, the adaptation of the system to the users' changing interests and the uncontrolled growth of the number of past experiences. With these problems at hand, we start to develop a mechanism to control the relevance of experiences on a CBR basis.

If, on the basis of our experience, we explore the use of AI techniques (i.e., CBR) in recommender systems, it is true that we would be also convinced that the system engineering insights could be also transferred to AI. In this sense, we know that systems can be controlled and become stable after a transition phase. At the stable position, the response of the system cannot be improved unless some perturbation is applied to the system. Such perturbation is an unexpected signal from outside the system and its control. We then analyse a similar behaviour in recommender systems: after an initial phase, the system becomes stable as much as it knows the preferences of the user, up to some degree, and the recommendations generated are always of the same kind and with the same degree of success. Then, in order to improve recommender systems, we believe that some inputs outside the system and its user should be incorporated as a perturbation. Analysing such situation, we found that AI researchers have focussed on the development of collaborative recommender systems that can recommend items to a user based on information about other users. Collaborative recommender systems allow the collaboration among users in a selection process, as in real life happens. Society in general, but specially our friends,

help us to find new amazing things. Often our friends tell us about an interesting product, movie, book or restaurant, helping us in the selection process.

In a collaborative environment, social patterns should be also modeled together with the particular user preferences. Under such perspective, *intelligent agents* [Maes 94] provide users with a means for managing information in a rational way. Agent properties such as encapsulation or social ability make agents suitable for dealing with the recommendation task. The encapsulation of user's personal information allows recommender agents to maintain privacy, while the social ability provides agents with a mechanism to collaborate, transforming recommender systems to *multi-agent systems of collaborative recommender agents* [Klusch 01].

Collaborative recommender agents take advantage of the collaborative world in which users can share information about their interests and preferences in order to find similar users who can help them. However, a serious trade-off between privacy and collaboration should be generated. That is, the broadcasting of personal information raises privacy concerns. In this thesis, we also define an alternative to deal with this trade-off and we propose a mechanism of collaboration based on *trust* [March 94] so that users maintain their privacy while benefiting from personalised collaboration. Trust techniques have proved useful in making agents less vulnerable to others, a fundamental need in open environments such as the Internet.

1.2 Objectives

The objectives of this thesis can be encompassed in four general purposes: the study of existing recommender systems, the application of CBR to recommender systems, the design of a mechanism of collaboration among users and the evaluation of recommender systems.

In a relatively short time, many recommender systems have been developed on the Internet, all of which take advantage of a particular set of AI techniques. The first general purpose of this thesis consists of bibliographic research into these systems. The study of the different AI techniques used to recommend to the user will provide us with a general view of recommender systems. We should then be able to generate a taxonomy that classifies the different techniques. The dimensions of this taxonomy will encompass how the information about the user is stored and how this

information can be used in order to recommend to the user.

According to our experience, CBR seems a suitable technique to be applied to recommender systems. Thus, our second general purpose consists of the adaptation of the CBR methodology to the recommendation field. We also want to propose a "forgetting mechanism", based on the human brain, as a technique to control the relevance of past experiences and forget the irrelevant ones. The aim of this mechanism is to better adapt the representation of the user's interests to the user and to control the uncontrolled growth of the past experiences stored.

Recommender systems often combine the information about the interests of the users in order to perform a collaboration among them. However, this requires the revelation of personal information. Our third general purpose is to design a mechanism of collaboration that maintains the privacy of user's personal data. In particular, we count on the utilisation of collaborative agents, since properties like their encapsulation and their social ability give us a suitable tool to achieve our purpose. When and with whom to collaborate are aspects to take into serious account. The concept of trust in other agents could provide us with a useful instrument to handle these issues. Thus, we will transform the typical centralised recommender system to a distributed world of recommender agents who collaborate by means of a mechanism based on trust.

Finally, our proposals will be evaluated. To date, it is very difficult to determine how well recommender systems work, since this involves purely subjective assessments. Our final purpose is to design an evaluation procedure for recommender systems as similar as possible to an evaluation performed with real users. Such a procedure should be based on the combination of information about real users and a simulator. Thanks to this simulator, we will be able to carry out repeatable and perfectly controlled experiments in order to show the performance of our CBR and collaborative approaches to recommender systems.

1.3 Outline of the Thesis

This document is structured in 5 chapters with a bibliography section at the end.

Chapter 2 details the study of existing recommender systems. In particular, 37 recommender systems have been deeply analysed. This analysis has resulted in

the identification of 8 dimensions in which recommender systems can be classified. These dimensions, together with their possible values, constitute a taxonomy of recommender systems. Thus, this chapter provides us with a state-of-the-art description of recommender systems; indispensable information before designing our proposals.

Chapter 3 describes our CBR approach to recommender systems. The different phases of the CBR cycle have been redefined in order to adapt the methodology to the recommendation task. A "forgetting mechanism" is then proposed in order to better adapt case-based profiles to the user's interests and to solve the utility problem of CBR systems. In order to evaluate this proposal, an evaluation method, which we call the *profile discovering procedure*, is proposed. This method simulates the recommendation process from information about real user's interests. Thanks to the profile discovering procedure, the results of our proposal to CBR recommender system are obtained and analysed in this chapter.

Chapter 4 presents our proposal of collaborative recommender agents. Users have a personal agent in charge of recommending interesting items to them. In order to collaborate, recommender agents exchange information by means of the opinion-based filtering method, which is based on a model of trust. Moreover, the typical collaborative filtering method is improved by means of this model. Both methods and the trust model are thoroughly explained in this chapter. Then, this proposal is evaluated through an extension of the profile discovering procedure that contemplates the collaboration between users.

Finally, chapter 5 concludes this document restating the main contributions of this thesis and listing further work. A list of related publications and prizes is included.

Chapter 2

Survey On Recommender Systems

Recently, Artificial Intelligence techniques have proved useful in helping users to handle the large amount of information on the Internet. The idea of personalised search engines, intelligent software agents, and recommender systems has been widely accepted among users who require assistance in searching, sorting, classifying, filtering and sharing this vast quantity of information. In this chapter, we present a state-of-the-art taxonomy of recommender systems on the Internet. We have analysed 37 different systems and their references and have sorted them into a list of 8 basic dimensions. These dimensions are then used to establish a taxonomy under which the systems analysed are classified. Finally, we conclude this chapter with a cross-dimensional analysis with the aim of providing a starting point for researchers to construct our own recommender system.

2.1 Introduction

In this chapter, we carry out a comprehensive and systematic study of recommender systems [Sanguesa 00]. Analysing the different systems, we have identified a list of 8 dimensions in which recommender systems can differ and possible values for these dimensions, therefore providing a taxonomy.

The intention of this chapter is to present state-of-the-art elements organised into a simple classification, explain the methods used and describe their advantages and disadvantages. Thus, the main purpose is to provide a starting point for researchers to construct their own recommender systems.

NAME	REFERENCES	DOMAIN
ACR News	[Mobasher 00]	Netnews Filtering
Amazon	[Amazon 03]	E-commerce
Amalthaea	[Moukas 97]	Web Recommender
Anatagonomy	[Sakagami 97]	Personalised Newspaper
Beehive	[Huberman 96]	Sharing News
Bellcore Video Recommender	[Hill 95]	Movie Recommender
Casmir	[Berney 99]	Document Recommender
CDNow	[CDNow 03]	E-commerce
Fab	[Balabanovic 97]	Web Recommender
GroupLens	[Resnick 94]	Netnews Recommender
ifWeb	[Minio 96, Asnicar 97]	Web Recommender
InfoFinder	[Krulwich 95, Krulwich 96]	Information Recommender
INFOrmer	[Riordan 95, Sorensen 97]	Netnews Filtering
Krakatoa Chronicle	[Kamba 95]	Personalised Newspaper
LaboUr	[Schwab 01]	Document Recommender
Let's Browse	[Lieberman 99]	Web Recommender
Letizia	[Lieberman 95]	Web Recommender
LifeStyle Finder	[Krulwich 97]	Purchase, Travel and Store Recommender
MovieLens	[Good 99]	Movie Recommender
News Dude	[Billsus 99]	Netnews Recommender
NewsWeeder	[Lang 95]	Netnews Recommender
NewT	[Sheth 93]	Netnews Filtering
Personal WebWatcher	[Mladenic 96]	Web Recommender
PSUN	[Sorensen 95]	Netnews Recommender
Re:Agent	[Boone 98]	E-mail Filtering
Recommender	[Basu 98]	Movie Recommender
Ringo/FireFly	[Shardanand 94, Shardanand 95]	Music Recommender
SIFT Netnews	[Yan 95]	Netnews Filtering
SiteIF	[Stefani 98]	Web Recommender
Smart Radio	[Hayes 99, Hayes 00]	Music Lists Recommender
Syskill&Webert	[Pazzani 96, Pazzani 97]	Web Recommender
Tapestry	[Goldberg 92]	E-mail Filtering
Webmate	[Chen 98]	Web Recommender
WebSail	[Chen 00]	Web Search Filtering
WebSell	[Cunningham 01]	Purchase Recommender
Websift	[Cooley 99]	Web Recommender
WebWatcher	[Armstrong 95, Joachims 97]	Web Recommender

Table 2.1: Domain of the Analysed Systems.

This chapter is organised as follows. First, we present the dimensions that constitute the taxonomy, which we group in two blocks: dimensions regarding profile generation and maintenance and dimensions related to profile exploitation. In sections 2.3 and 2.4 we provide the classification of the systems according to dimensions of profile generation and maintenance and profile exploitation respectively. We continue by performing a cross-dimensional analysis in section 2.5 and end with section 3.7 in which several conclusions are presented.

2.2 The Taxonomy

In a relatively short time, several recommender systems have been developed and there is a wide variety of such systems, all of which take advantage of a particular set of AI techniques. We have followed two main approaches in this study of recommender systems: spatial and functional. The spatial approach produces a classification of systems according to the application domain (see Table 2.1 for the domains of the various systems analysed). The functional approach produces a classification based on the different task-achievement techniques used in the system. This latter approach allows us to study the systems systematically and consequently, we have spent more time on it.

Consistently, when analysing how a recommender system makes recommendations or assesses a user, the key issue is the user profile. User profile generation and maintenance requires five design decisions which constitute the first five dimensions of our taxonomy: the profile representation technique, the technique used to generate the initial profile, the source of the relevance feedback which represents the user interests, the profile learning technique and the profile adaptation technique. Figure 2.1 shows the relationships between these techniques in the generation and maintenance of user profiles.

The *profile representation* is the first step to take into account in a recommender system since the other techniques depend on it. Once this step is decided, the other techniques can be defined. A recommender system cannot begin to function until the user profile has been created. Furthermore, the system needs to know as much as possible from the user in order to provide him/her with satisfactory results from the very beginning. Therefore, systems need to use a suitable technique in order to

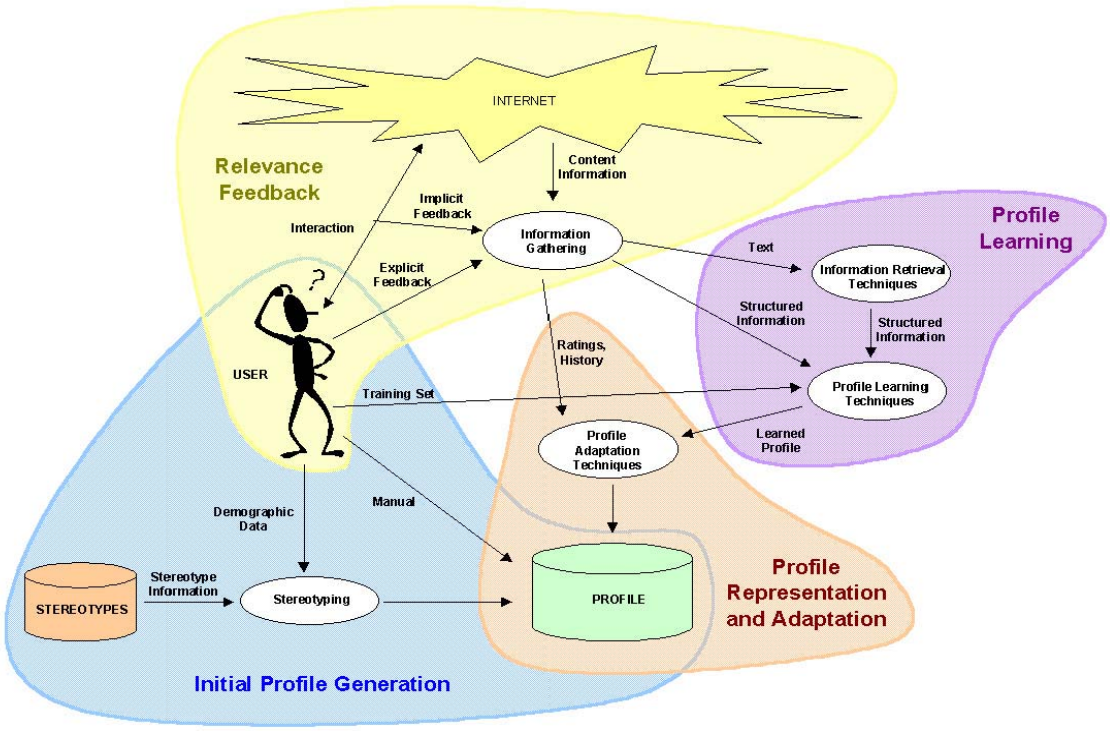


Figure 2.1: Profile Generation and Maintenance.

generate an accurate *initial profile*.

To generate and maintain the user profile, the system needs relevant information about the user's interests. When users interact with a computer, they provide a great deal of information about themselves. Successful interpretation of these data streams is necessary for computers to tailor themselves to each individual's behaviour, habits and knowledge. As for the interaction of the user with these applications, the system can gather relevance feedback to learn his tastes, interests and preferences. *Relevance feedback* is then a main dimension for recommender systems. Typically, the feedback, given explicitly or implicitly by the user, has no sense in itself. Therefore, there is a need for a *profile learning technique* which extracts the relevant information and structures this information depending on the representation of the profile.

User tastes usually change as time goes on. Therefore, the user profile should also be changed in order to retain the desired accuracy in its exploitation. Hence, the need for a technique to *adapt the user profile* to new interests and to forget old ones.

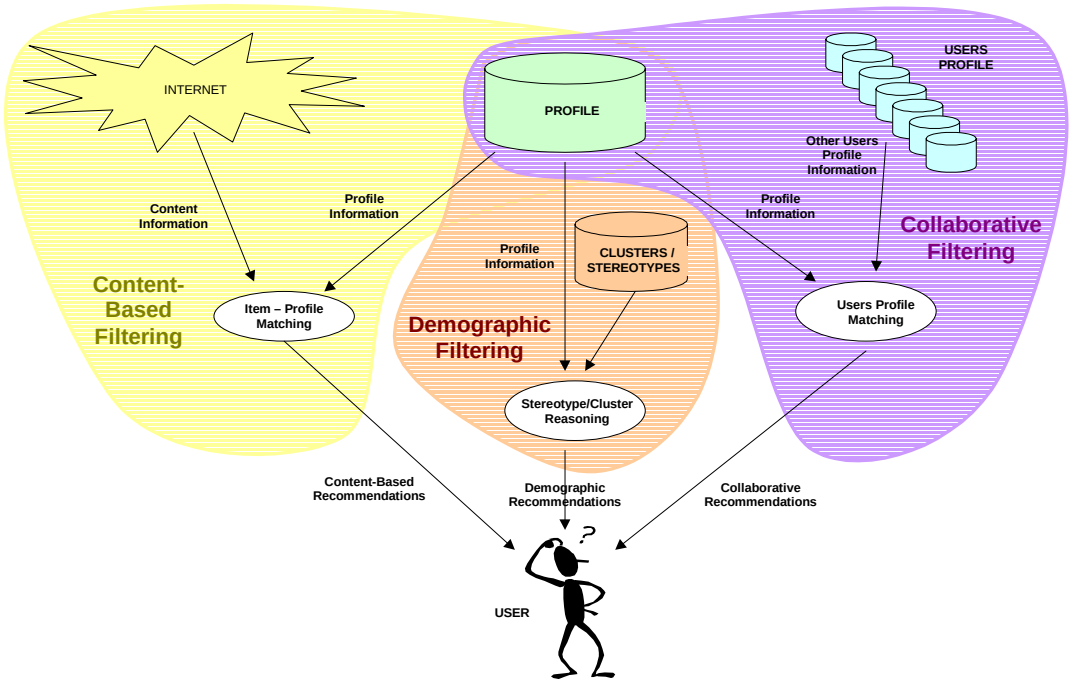


Figure 2.2: Profile Exploitation for Recommendation.

Once there is a user profile available, recommender systems exploit it to recommend either products or actions to a user. Recommender systems make decisions according to the information available. Such information includes data about items as well as different profiles of other users on the web. Since there is so much information, a fundamental issue is to select the most appropriate information with which to make decisions. In other words, an information filtering method is essential. There are three information filtering approaches for making recommendations: demographic filtering, content-based filtering and collaborative filtering. Demographic filtering uses descriptions of people to learn the relationship between a particular item and the type of people who like it. Content-based filtering uses descriptions of the content of the items to learn the relationship between a single user and the description of the items. Several *user profile-item matching* methods can be used in order to compare the user's interests and the items. Collaborative filtering uses the feedback from a set of people concerning a set of items in order to make recommendations, but ignores the content of the items. Various methods are used by the systems to *match user profiles* and find users with similar interests.

TAXONOMY OF RECOMMENDER SYSTEMS	
Profile Generation and Maintenance	**Profile Exploitation**
User Profile Representation	Information Filtering Method
Initial Profile Generation	User Profile-Item Matching Technique
Profile Learning Technique	User Profile Matching Technique
Relevance Feedback	
Profile Adaptation Technique	

Table 2.2: Dimensions of the Taxonomy.

In terms of profile exploitation, then, three main dimensions characterise recommender systems: the information filtering method (demographic, content-based and collaborative), the item-profile matching (when content-based) and the user profile matching techniques (when collaborative). See Figure 2.2 for a general view.

All in all, we have identified, from a functional viewpoint, 8 classification dimensions for recommender systems, 5 in terms of profile generation and maintenance and 3 in terms of profile exploitation (see Table 2.2). We will now go on to discuss these in further detail.

2.3 Profile Generation and Maintenance

Five design decisions should be taken to generate and maintain a user profile: the representation, the technique to generate the initial profile, the source of the relevance feedback which represents the user interest, the profile learning technique and the profile adaptation technique (see Figure 2.1).

2.3.1 User Profile Representation

Constructing accurate profiles is a key task since the system's success will depend, to a large extent, on the ability to represent the user's current interests. Accurate profiles are vital for both the content-based component (to insure recommendations are appropriate) and the collaborative component (to insure that users with similar profiles are indeed similar).

Several approaches have been taken to represent user profiles, such as a history of purchases, web navigation or e-mails, an indexed vector of features, a n-gram, a semantic network, an associative network, a classifier including neural networks,

NAME	TECHNIQUE
ACR News	Frequent Itemsets, URL Clusters
Amazon	Purchase History with Ratings
Amalthaea	Weighted Feature Vector
Anatagonomy	Weighted Feature Vector
Beehive	Clusters(Weighted Feature Vector)
Bellcore Video Recom	User-Item Ratings Matrix
Casmir	Weighted Feature and Document Network
CDNow	Purchase History with Ratings
Fab	Weighted Feature Vector
GroupLens	User-Item Ratings Matrix
ifWeb	Multivalued Weighted Attributes, Weighted Semantic Network
InfoFinder	Decision Tree
INFOrmer	Weighted Associative Network
Krakatoa Chronicle	Weighted Feature Vector
LaboUr	Probabilistic Feature Vector, Boolean Feature Vector
Let's Browse	Weighted Feature Vector
Letizia	Weighted Feature Vector
LifeStyle Finder	Demographic Features
MovieLens	Weighted Feature Vector, Inducted Rules
News Dude	Short Term: Weighted, Long Term: Boolean Feature Vector
NewsWeeder	Weighted Feature Vector
NewT	Weighted Feature Vector
Personal WebWatcher	Probabilistic Feature Vector
PSUN	Weighted N-Grams
Re:Agent	Weighted Feature Vector, Neural Network
Recommender	Inducted Rules
Ringo / FireFly	User-Item Ratings Matrix
SIFT Netnews	Boolean Feature Vector, Weighted Feature Vector, Decision Tree
SiteIF	Weighted Semantic Networks
Smart Radio	User-Item Ratings Matrix
Syskill & Webert	Probabilistic Feature Vector, Boolean Feature Vector, Decision Tree, Weighted Feature Vector
Tapestry	Indexed Messages and Annotations
Webmate	Weighted Feature Vector
WebSail	Boolean Feature Vector
WebSell	Interesting/Not Interesting Products
Websift	Inducted Rules, Patterns, Statistics
WebWatcher	Boolean Feature Vector

Table 2.3: Profile Representation Technique of the Systems.

decision trees, inducted rules or Bayesian networks, a matrix of ratings and a set of demographic features. Table 2.3 shows the user profile representation techniques used by the different systems analysed.

History-Based Model

Some systems keep a list of purchases, the navigation history in WWW or the content of e-mail boxes as a user profile. Additionally, it is also common to keep the relevant feedback of the user associated with each item in the history.

A historical approach is most commonly used in e-commerce, in which systems keep a list of purchased products and user ratings, as a user profile. This is the

case in the two most popular state-of-the-art recommender systems in e-commerce: Amazon.com [Amazon 03] and CDNow.com [CDNow 03]. A similar approach is used in WebSell [Cunningham 01], in which the profile is defined by using two lists, one with purchased products rated as *interesting* and another with *uninteresting* ones. Another approach is implemented in Tapestry [Goldberg 92], an e-mail filtering system which builds a profile while keeping track of messages and annotations given by the user.

Vector Space Model

In the vector space model, items are represented with a vector of features, usually words or concepts, with an associated value. This value can be a Boolean or a real number. The Boolean value represents the presence of the value of the feature, and the real number represents the frequency, relevance or probability of the feature, which is calculated using information indexing techniques (see section 2.3.3).

For example, Webmate [Chen 98] utilises a multiple feature vectors representation. The basic idea is to represent each document as a vector in a vector space so that documents with similar content have similar vectors. Each dimension of the vector space represents a word and its weight, calculated as a combination of the statistics term frequency (see section 2.3.3).

Weighted N-Grams

In weighted n-grams, items are represented with a net of words with weights in the nodes and edges. For example, PSUN [Sorensen 95], based on the assumption that words tend to occur one after another a significantly high number of times, extracts fixed length consecutive series of n characters and organises them with weighted links representing the co-occurrence of different words. Therefore, the structure achieves a context representation of the words.

Weighted Semantic Networks

Semantic networks [Potter 88] are able to store the meanings of words, so the human-like use of these meanings is possible. Minio and Tasso, in the ifWeb system [Minio 96], implement such an approach. In IfWeb, a semantic network base

contains a collection of semantic networks describing a typical pattern of topics of interest to the user. The Stefani and Strapparava approach in the SiteIF system [Stefani 98] represents every node as a word or an interesting concept and the arcs between nodes are the co-occurrence relation between two words; every node and every arc has a weight representing a different level of interests to the user.

Weighted Associative Networks

An associative network consists of a set of nodes which represent primary terms, concepts or words, in which a user is interested. A set of weighted links establishes the organisation of these terms into relevant phrases. Associative networks differ from the semantic networks because semantic networks have different generic link types such as synonymy, superclass-subclass, and also possibly disjunctive and conjunctive sets of links. In contrast, associative networks have only a single link type, a weighted edge, the semantics being implicit in the structure of the network and the parameters associated with the processing [Riordan 95].

Classifier-Based Models

Systems using a classifier as a user profile learning technique retain the structure of the classifier as a profile. This is the case in neural networks, decision trees, inducted rules and Bayesian networks.

A neural network is a network of input and output cells, based upon neuron functions in the brain. Neural networks create a compact representation that responds to queries quickly. However, they can be slow to train. For example, Re:Agent [Boone 98] filter e-mails through a neural network previously trained with feature vectors of past messages.

A decision tree is another way to classify data. It consists of a set of nodes and a set of directed edges that connect the nodes (tree structure). The internal nodes represent questions about the parameters, and the edges represent answers to those questions, i.e. values for the parameters. The leaf nodes represent a final decision. For example, InfoFinder [Krulwich 96] recommends documents based on a decision tree.

Association rules have been used for many years in merchandising, both to anal-

yse patterns of preference across products, and to recommend products to consumers based on other products they have selected. Association rules can form a compact representation of preference data which improve efficiency of storage as well as performance. For example, an association rule expresses the relationship that a certain movie is often purchased along with others [Basu 98].

A Bayesian network is a directed acyclic graph in which nodes represent propositional variables and arcs represent dependencies [Jensen 96]. A node's value is a function of the values of the nodes it depends upon. Leaf nodes represent propositions, which can be determined by observation. The resulting model is very small, very fast and essentially as accurate as nearest neighbours methods [Breese 98].

User-Item Ratings Matrix

Some collaborative filtering systems maintain a user-item ratings matrix as a user profile. The user-item ratings matrix contains historical user ratings on items. Each cell (u, i) of the matrix contains a rating representing the evaluation of the user u to the item i, and an empty value if there is no evaluation.

These systems do not use any learning profile technique (see section 2.3.3) but bring together all the processes in the user profile matching techniques (see section 2.4.3).

Demographic Features

Demographic filtering systems create a user profile through stereotypes. Therefore, the user profile representation is a list of demographic features which represent the kind of user. None of these systems use any learning profile technique (see section 2.3.3) but they bring together all the processes in stereotype reasoning [Kobsa 01].

2.3.2 Initial Profile Generation

It is desirable to learn as much as possible from the user so that the recommender systems provide satisfactory results from the very beginning. However, the user is not usually willing to spend much time in defining his interests to create his profile.

NAME	TECHNIQUE
ACR News	Training Set
Amalthaea	Manual
Amazon	Empty
Anatagonomy	Empty
Beehive	Empty
Bellcore Video Recom	Training Set
Casmir	Unknown
CDNow	Empty
Fab	Empty
GroupLens	Empty
ifWeb	Training Set, Stereotyping
InfoFinder	Training Set
INFOrmer	Training Set
Krakatoa Chronicle	Empty
LaboUr	Training Set
Let's Browse	Training Set
Letizia	Empty
LifeStyle Finder	Stereotyping
MovieLens	Training Set
News Dude	Training Set
NewsWeeder	Training Set
NewT	Training Set
Personal WebWatcher	Manual
PSUN	Training Set
Re:Agent	Manual, Training Set
Recommender	Training Set
Ringo / FireFly	Training Set
SIFT Netnews	Training Set
SiteIF	Empty
Smart Radio	Training Set
Syskill & Webert	Manual, Stereotyping
Tapestry	Empty
Webmate	Empty
WebSail	Empty
WebSell	Empty
Websift	Training Set
WebWatcher	Manual

Table 2.4: Initial Profile Generation Technique of the Systems.

Moreover, users' interests may change over time, making the profiles difficult to maintain. For these reasons, starting up and maintaining user profiles is a difficult aspect in the design and development of recommender systems. The degree of automation in the acquisition of user profiles can range from manual input, to semi-automatic procedures (stereotyping and training sets), to the automatic recognition by the systems themselves. Table 2.4 shows the initial profile generation techniques used by the different systems analysed.

Empty

Some systems do not bother with the initial profile and start with an empty profile structure (e.g., [Chen 98, Balabanovic 97, Cunningham 01]). There is no initial

phase, the profile structure is filled through an automatic recognition method when the user begins interaction with the system.

Manual

A manual system asks users to register their interests in the form of keywords, topics and so on. One of the advantages of this method is the transparency of the system behaviour. When items have been delivered to a user, he/she can usually easily guess why each item was delivered. One problem with this method though, is that it requires much effort on the part of the user. Another problem is that people cannot necessarily specify what they are interested in, because their interests are sometimes still unknown.

Stereotyping

Stereotyping is based on the fact that creating an initial model is, in a sense, a classification problem, aimed at generating initial predictions about the user [Kobsa 01]. The user model is initiated by classifying users in stereotypical descriptions [Rich 79], representing the features of classes of users. The use of stereotypes in computer systems for maintaining models of their users was introduced by Rich with the Grundy system. Typically, the data used in the classification is demographic and the user is asked to fill out a registration form: record data (name, address, etc.), geographic data (area code, city, etc), user characteristics (age, sex, etc.), psychographic data (e.g., lifestyle), etc.

An example is the method implemented by Krulwich in the LifeStyle Finder [Krulwich 97] which uses a commercially available database of demographic data which encompasses the interests of people nationwide.

The main shortcoming of this technique is the difficulty of acquiring personal data from the users. Internet users normally avoid engaging in a relationship with Internet sites. This is mostly due to a lack of faith in the privacy policy of today's web sites. Normally, users either withhold personal data or provide false data.

Training Set

The training set is a collection of user interaction examples which is used to infer the initial user profile. One practical way to establish the training set is to ask the user to rate some concrete examples as relevant or irrelevant to their interests (e.g., [Sorensen 95] and [Boone 98]). A similar approach is to ask the user to rate a set of predefined examples (e.g., [Good 99] and [Shardanand 94]). In both cases, once the user has given the appropriate information, the system processes the data with one of the learning techniques explained in section 2.3.3.

This mode has the advantage of simplified handling. It has the disadvantage, and the danger, that someone has to select the examples which are not always representative and the results are less precise. Some of the systems using this technique are ACR News [Mobasher 00], FireFly [Shardanand 95] and LaboUr [Schwab 01].

2.3.3 Profile Learning Techniques

The previous section described sources of information potentially representative of user interests, mainly the training sets. Profile learning techniques build a user profile through these data. These techniques can be seen as a preliminary step in representing a user profile.

It is important to note that when the learning data is composed of text without structure, it is necessary to pre-process the information in order to get structured, relevant information. Some systems simply use an information indexing technique to build a profile and represent it as a structure of indexed words, although information indexing techniques cannot be considered artificial intelligence techniques.

Some systems have an off-line phase during which they learn a model of a user behaviour, and then an online phase during which they apply the model in real time. Most systems, however, use a lazy learning approach (online), in that they build and update the model while making recommendations in real time. Off-line learning methods may prove practical for environments in which knowledge of consumer preferences changes slowly with respect to the time needed to build the model, but they are not suitable for environments in which consumer preference models must be updated rapidly or frequently.

In this section, we will first briefly explain some typical systems for which profile

NAME	TECHNIQUE
ACR News	Induction Rule Learning, Clustering
Amazon	Not Necessary
Amalthaea	Feature Selection(Stemming), Information Indexing (TF-IDF)
Anatagonomy	Information Indexing(TF-IDF)
Beehive	Clustering
Bellcore Video Recom	Not Necessary
Casmir	Simple Positive Reinforcement, Simple Positive Reinforcement with Query Keyword Overriding, Positive and Negative Reinforcement, Positive and Negative Reinforcement with Query Keyword Overriding
CDNow	Not Necessary
Fab	Information Indexing(TF-IDF)
GroupLens	Not Necessary
ifWeb	Feature Selection(Stop-Words, Stemming, ...)
InfoFinder	Feature Selection(Heuristics), Decision Tree(ID3)
INFOrmer	Feature Selection(Stop-Words, Stemming, ...)
Krakatoa Chronicle	Information Indexing(TF-IDF)
LaboUr	Information Indexing(Boolean)
Let's Browse	Information Indexing(TF-IDF)
Letizia	Information Indexing(TF-IDF)
LifeStyle Finder	Not Necessary
MovieLens	Information Indexing(TF-IDF), Induction Rule Learning (Ripper)
News Dude	Short Term: Information Indexing(TF-IDF), Information Indexing(Boolean)
NewsWeeder	Information Indexing(TF-IDF), MDL
NewT	Feature Selection(Stop-Words, Stemming), Information Indexing(TF-IDF)
Personal WebWatcher	Information Indexing(TF-IDF)
PSUN	Feature Selection(Stemming), N-Gram Induction
Re:Agent	Feature Selection(Stop-Words), Information Indexing(TF-IDF), Clustering, Neural Network
Recommender	Induction Rule Learning (Ripper)
Ringo / FireFly	Not Necessary
SIFT Netnews	Information Indexing(Boolean), Information Indexing(TF-IDF)
SiteIF	Feature Selection(Stop-Words, Stemming, ...)
Smart Radio	Not Necessary
Syskill & Webert	Feature Selection(Stop-Words), Information Indexing(Boolean), Information Indexing(TF-IDF), Decision Tree(ID3)
Tapestry	Not Necessary
Webmate	Information Indexing(TF-IDF)
WebSail	Information Indexing(TF-IDF)
WebSell	Not Necessary
Websift	Induction Rule Learning
WebWatcher	Information Indexing(TF-IDF), Winnow, WordStat, Random

Table 2.5: Profile Learning Technique of the Systems.

learning techniques are not necessary. Then, we summarise the information retrieval techniques used to preprocess information. Finally, we look at the most commonly used profile learning techniques are reviewed: clustering and classifiers. Table 2.5 shows the profile learning techniques used by the different systems analysed.

Not Necessary

Some systems keep information directly acquired from the system as a user profile, therefore, they do not need a profile learning technique. There are three main kinds

of systems do not need a profile learning method:

- Systems which acquire user profile information from a database. For instance, electronic commerce systems ([Amazon 03, CDNow 03, Cunningham 01]) which extract the information from a database of products and keep a purchase list as a profile (see section 2.3.1).
- Collaborative filtering systems ([Goldberg 92, Resnick 94, Shardanand 95]) which keep a matrix with the user-item ratings as a profile (see section 2.3.1).
- Systems which create an initial profile through stereotyping (see section 2.3.2) and do not modify it ([Krulwich 97]). This is the case in demographic filtering systems (see section 2.4.1).

Systems that do not need a profile learning technique concentrate information filtering tasks on the profile-item (see section 2.4.2) or profile-profile (see section 2.4.3) matching techniques.

Structured Information Retrieval Techniques

When information has no structure (e.g. text), some kind of pre-processing step is needed to extract structured relevant information. Typically, this process comprises two main steps: feature selection and information indexing.

Feature selection can be achieved through different approaches which reduce the number of words: stop-words, pruning, stemming, etc (see [Salton 83]).

Information indexing uses the frequency word occurrence to calculate the potential relevance of an item. TF-IDF is one of the most successful and best-tested techniques. A document is represented as a vector of weighted terms (see section 2.3.1). The computation of the weights reflects empirical observations regarding text. Terms frequently appearing in a document (TF=term-frequency), but rarely on the outside (IDF=inverse-document-frequency), are more likely to be relevant to the topic of the document.

TF-IDF has two popular variants: the boolean method and the probabilistic method. The boolean method is a simplistic approach where the profile is represented as a vector of words with a boolean value indicating their presence in the text.

The probabilistic method is a generalisation of the exact match technique. In this method, documents are ranked by the probability that they satisfy the information need rather than by making a shape decision. Bayesian inference networks have proven to be a useful technique for computing this probability (see section 2.3.3).

Clustering

The basic idea of this technique is clustering similar user information into groups based on data. Then, these clusters are matched against actual information to conclude whether it is interesting or not.

For example, in ACR News [Mobasher 00] user transactions are mapped into a multi-dimensional space as vectors of URL references. This space is partitioned into clusters (*usage clusters*) representing a group of transactions that are similar, based on co-occurrence patterns of URL references. Finally, clusters are matched against an active user session to recommend interesting URLs.

Traditional collaborative filtering techniques are often based on matching the current user profile against clusters of similar profiles obtained by the system over time from other users (see section 2.4.3).

Classifiers

Classifiers are general computational models for assigning a category to an input. To build a recommender system using a classifier means using information about the item and the user profile as input, and having the output category represent how strongly to recommend an item to the user. Classifiers may be implemented using many different machine learning strategies including neural networks, decision trees, association rules and Bayesian networks.

Learning in neural networks is achieved by training the network with a set of data. Each input pattern is propagated forward through the network and active output cells represent the interest of the user. When an error is detected it is propagated backward adjusting the cell parameters to reduce the error, thus achieving learning. For instance, Jennings and Higuchi employed a neural network for constructing a user's profile [Jennings 93].

Decision tree learning is a method for approximating discrete-valued target func-

tions, in which the learned function is represented by a decision tree. The learned trees can also be represented as a set of if-then rules. Decision tree learners build a decision tree by recursively partitioning examples into subgroups, obtaining source classes of items which can be classified, for example, into *interesting* and *not interesting* [Krulwich 95]. The most widely-used decision tree learner applied to profiling is the ID3 [Quinlan 83].

The discovery of association rules by inductive learning [Mobasher 00] is one of the best-known classifier examples. The association rule discovery methods initially find groups of items occurring frequently together in many transactions. Such groups of items are referred to as frequent item sets. Association rules capture the relationships among these items based on their patterns of co-occurrence across transactions. Association rules can form a very compact representation of preference data which may improve efficiency of storage as well as performance. Some examples of inductive learning techniques are Ripper [Cohen 95], Slipper [Cohen 99], CN2 [Clark 89] and C4.5rules [Quinlan 94].

A Bayesian network learner algorithm is applied to a set of training data, searching over various model structures in terms of dependencies for each item. In the resulting network, each item will have a set of parent items that are the best predictors of its votes. The model can be built off-line in a matter of hours. Thus, this technique may prove practical for environments in which knowledge of consumer preferences changes slowly with respect to the time needed to build the model.

2.3.4 Relevance Feedback

Human interests change as time passes. For example, a new father may be interested in infant care just after childbirth, but this interest gradually decreases over time. Therefore, the recommender system needs up-to-date information to update the user profile automatically. In this section, several ways to obtain this information, which we call relevance feedback, are presented. Then, in section 2.3.5 we will see how to use this information to update user profiles.

Typically, it is possible to distinguish two kinds of relevance feedback: positive information (items liked by the user) and negative information (i.e., inferring features which the user is not interested in [Holte 96]). The authors claim that negative information produces a dramatic improvement in the system's performance. How-

ever, there are a few systems which cannot take into account negative information because the system's accuracy is likely to decrease (e.g., [Schwab 00]). So, it all depends on the system.

The two most common ways to obtain relevance feedback is to use information given explicitly or to get information observed implicitly from the user's interaction. Moreover, some systems propose implicit/explicit hybrid approaches. Table 2.6 shows the relevance of feedback techniques used by the different systems analysed.

No Feedback

Some systems do not update the user profile automatically and, therefore, they do not need relevance feedback. For example, all the systems which update the user profile manually (see section 2.3.5). Of course, systems which never modify the profile do not need relevance feedback either.

For instance, SIFT Netnews [Yan 95] creates an initial profile of the user and does not update it automatically over time. However, the user can modify his profile manually.

Explicit Feedback

In several systems, users are required to explicitly evaluate items. These evaluations indicate how relevant or interesting an item is to the user, or how relevant or interesting the user thinks an item is to other users [Rich 79].

There are three main approaches to get explicit relevance feedback: like/dislike (e.g., [Chen 00, Billsus 99]), ratings (e.g., [Shardanand 95, Moukas 97]) and text comments (e.g., [Resnick 94, Goldberg 92]). In like/dislike systems, users are required to explicitly judge items on a binary scale, i.e., classify an object as *"interesting"* or *"not interesting"*, as *"relevant"* or *"not relevant"* or as *"like"* or *"hate"*. The ratings approach requires users to provide a judgement on a discrete scale. The rating scale is typically numeric (e.g., the web bookstore Amazon.com offered users the opportunity of rating books in various categories on a 5-point scale) or symbolic with mapping to a numeric scale (e.g., in Syskill&Webert [Pazzani 96] users have the possibility of rating a Web page as *"hot"*, *"lukewarm"*, or *"cold"*). Finally, several sites encourage textual comments from their users (e.g., Grouplens [Resnick 94]

NAME	TECHNIQUE
ACR News	Implicit(Navigation History)
Amazon	Explicit(Ratings), Implicit(Purchase History)
Amalthaea	Explicit(Ratings)
Anatagonomy	Explicit(Ratings), Implicit(Scrolling, Enlarging)
Beehive	Implicit(Mail History)
Bellcore Video Recom	Explicit(Ratings)
Casmir	Explicit(Ratings)
CDNow	Explicit(Ratings), Implicit(Purchase History)
Fab	Explicit(Ratings)
GroupLens	Explicit(Ratings, Text Comments), Implicit(Time Spent)
ifWeb	Explicit(Ratings)
InfoFinder	Explicit(Ratings)
INFOrmer	Explicit(Ratings)
Krakatoa Chronicle	Explicit(Ratings), Implicit(Saving, Scrolling, Time Spent, Maximising, Resizing, Peeking)
LaboUr	Implicit(Links, Time Spent)
Let's Browse	Implicit(Links, Time Spent)
Letizia	Implicit(Links, Time Spent)
LifeStyle Finder	Explicit(Ratings), Implicit(Purchase History)
MovieLens	Explicit(Ratings)
News Dude	Explicit(Like/Dislike, I already know this, Tell me more)
NewsWeeder	Explicit(Ratings)
NewT	Explicit(Like/Dislike)
Personal WebWatcher	Implicit(Links)
PSUN	Explicit(Ratings)
Re:Agent	No Feedback
Recommender	Explicit(Ratings)
Ringo / FireFly	Explicit(Ratings)
SIFT Netnews	Explicit(Like/Dislike)
SiteIF	Implicit(Links)
Smart Radio	Explicit(Ratings), Implicit(Saving)
Syskill & Webert	Explicit(Ratings)
Tapestry	Explicit(Like/Dislike, Text Comments), Implicit(Forwarding)
Webmate	Explicit(Like/Dislike)
WebSail	Explicit(Like/Dislike)
WebSell	Explicit(Unknown)
Websift	Implicit(Navigation History)
WebWatcher	Explicit(Goal Reached), Implicit(Links)

Table 2.6: Relevance Feedback Technique of the Systems.

and Tapestry [Goldberg 92]). Systems gather comments about a single item and present them to the users as a means of facilitating the decision-making process. While textual comments are helpful, they require a fair amount of processing by the targeted user. Users must read text and interpret to what degree it is positive or negative.

Explicit feedback has the advantage of simplicity. Several papers have demonstrated the high performance of systems using explicit relevance feedback [Salton 90, Buckley 95]. However, in practical applications, explicit feedback has three serious drawbacks:

- First, the relevance of information is always relative to the changing information need of a user, and information relevance judgements of individual items are typically assumed to be independent when, in fact, they are not (e.g., the third article presented on the same topic may simply be rated lower because the first two items satisfied information need and the user is judging incremental relevance at this point).
- Another problem is that numeric scales may not be adequate for describing the reactions humans have to items.
- The last problem is that computer users do not supply enough feedback, particularly negative feedback. Pazzani et al. report that only 15% of users would supply feedback even though they were encouraged to do so [Pazzani 97]. Users are generally very reluctant to perform actions not directed towards their immediate goals if they do not receive immediate benefits, even when they would profit in the long run [Carroll 87].

Implicit Feedback

Implicit feedback means that the system automatically infers the user's preferences passively by monitoring the user's actions. Chatterjee et al. prove empirically in [Chatterjee 98] that the user's interests can be inferred from his behaviour. Their results are important because motivating web consumers to provide personal data in an explicit way is proving very difficult. Conclusions about the user's interests should therefore not rely on user explicit feedback very much, but rather take passive observations about users into account as far as possible.

Implicit feedback was defined some years ago by Rich [Rich 79], and the first system was implemented by Mitchell et al. [Mitchell 85]. Since then, many systems have implemented implicit user feedback in their approaches (e.g., [Stefani 98, Schwab 01]) and some systems have even combined it with explicit feedback (see hybrid approach in section 2.3.4).

Most implicit methods obtain relevance feedback by analysing the links followed by the user (e.g., [Lieberman 95, Mladenic 96]), a history of purchases (e.g., [Amazon 03, CDNow 03, Krulwich 97]), the navigation history (e.g., [Cooley 99, Mobasher 00]), e-mail boxes (e.g., [Huberman 96]) and the time spent on a particular web page (e.g., [Morita 94, Konstan 97, Kobsa 01, Sakagami 97]).

There are many other examples of confirmatory actions. For documents like Web pages, news articles or e-mail messages, it is interesting to find out if the user takes any further processing action, such as saving a document ([Kamba 95]), printing a document, bookmarking a Web page, deleting a document, replying to or forwarding an e-mail [Goldberg 92], or scrolling, maximising, minimising or resizing the window containing the document or the Web page ([Kamba 95, Sakagami 97]). Since these actions are performed under the control of the application, they can be registered and evaluated to learn the user's profile.

However, Kobsa et al. [Kobsa 01] do not recommend a universal logging of usage data on the micro-interaction level, such as the tracking of mouse movements within applets, unless the purpose of the login has already been specified (e.g., for determining user's interest in page segments [Sakagami 97]). The amount of data collected is very large, the computation needed to derive recommendations for adaptations is extensive, and the confidence in the suitability of these adaptations is likely to be relatively low. However, experimentation with such data in smaller, laboratory contexts to drive the development of new methods in the area of implicit feedback seems promising.

Hybrid Approach

The limited evidence available on implicit feedback suggests that it has great potential but its effectiveness remains unproven. As is common in many technologies, the best performing system results in combining several existing technologies. In this field, implicit feedback can be combined with explicit feedback systems in a hybrid

system. Providing implicit feedback greatly decreases the user's efforts, whereas providing explicit feedback helps the system to infer user preferences accurately.

One approach with this combination involves using implicit data as a check on explicit ratings [Nichols 97]. For instance, if a user is explicitly rating an item, then there should be some implicit data to confirm that he has actually examined it. If there is no evidence to suggest this, then perhaps its rating should be ignored or reduced in importance. Conversely, an evaluation with a relatively long "examine time" may be increased in importance.

A different case is Anatagonomy [Sakagami 97]. Giving explicit feedback is optional and should only be used when users wish to show explicit interest. WebWatcher [Joachims 97], Krakatoa Chronicle [Kamba 95], GroupLens [Resnick 94], LifeStyle Finder [Krulwich 97], CDNow [CDNow 03] and Amazon [Amazon 03] also use hybrid relevance feedback.

2.3.5 Profile Adaptation Techniques

Since recommender systems typically involve interaction over long periods of time, user interests cannot be assumed to stay constant. This normally means that the most recent observations gathered through what we have called relevance feedback represent the user's current interests better than older ones. Therefore, there is a need for a technique that will adapt the user profile to new interests and forget old ones.

There are several approaches to this: manually (simply adding the new information), with a time window, aging examples, combining a short-term and a long-term model, a gradual forgetting function or the natural selection for ecosystems of agents. Table 2.7 shows the profile adaptation techniques used by the different systems analysed.

Manual

In some systems, the user has to change the profile when he/she is interested in updating it. For instance, in Sift Netnews [Yan 95], when the user wants to include/exclude one of the interests contained in his profile, he has to modify it manually.

NAME	TECHNIQUE
ACR News	Add New Information
Amalthaea	Natural Selection, Gradual Forgetting
Amazon	Add New Information
Anatagonomy	Add New Information
Beehive	Add New Information
Bellcore Video Recommender	Add New Information
Casmir	Add New Information
CDNow	Add New Information
Fab	Natural Selection
GroupLens	Add New Information
ifWeb	Gradual Forgetting
InfoFinder	Add New Information
INFOrmer	Add New Information
Krakatoa Chronicle	Add New Information
LaboUr	Gradual Forgetting
Let's Browse	Add New Information
Letizia	Add New Information
LifeStyle Finder	Add New Information
MovieLens	Add New Information
News Dude	Short-Term and Long-Term Models
NewsWeeder	Add New Information
NewT	Natural Selection
Personal WebWatcher	Add New Information
PSUN	Natural Selection
Re:Agent	Manual
Recommender	Add New Information
Ringo / FireFly	Add New Information
SIFT Netnews	Manual
SiteIF	Gradual Forgetting
Smart Radio	Add New Information
Syskill & Webert	Add New Information
Tapestry	Add New Information
Webmate	Add New Information
WebSail	Add New Information
WebSell	Add New Information
Websift	Add New Information
WebWatcher	Add New Information

Table 2.7: Profile Adaptation Technique of the Systems.

As in manual initial profile generation (see section 2.3.2), this approach has two important problems: it requires much effort on the part of the user and people cannot necessarily specify what they are interested in because their interests are sometimes still unknown. Therefore, manual updating turns out to be difficult when requirements change quickly.

Add New Information

This approach is the most commonly used in current systems. The idea is to update the user profile adding new information extracted from the user relevance feedback. Thus, the profile is adapted to the new user's interests. The main drawback, however, is that old interests are not forgotten.

Gradual Forgetting Function

The concept of gradual forgetting was introduced by Webb and Kuzmycz in [Webb 96] with the main idea being that natural forgetting is a gradual process. Therefore, a gradual forgetting function can be defined. It should produce a weight for each observation according to its location in the course of time. Webb and Kuzmycz suggest a data aging mechanism which places an initial weight of 1 on each observation. A set proportion discounts the weight of every observation each time another relevant observation is incorporated into the model. Thus, the most recent observations become more "important", assuming they better represent the current users' interests than the older ones. Hence, the system becomes more noise resistant without losing its sensitivity to real changes in interest [Schwab 01].

Koychev proposes a linear gradual forgetting function [Koychev 00], but it can be approximated with any function (e.g., logarithmic or exponential).

A particular case of the gradual forgetting function is the time window approach. It is the most frequently-used approach in dealing with the problem of forgetting old interests. It consists of learning the description of the user's interests from only the latest observations. The training examples are selected from a so-called time window, i.e. only the last examples are used for training [Mitchell 94]. An improvement on this approach is the use of heuristics to adjust the size of the window according to the current predictive accuracy of the learning algorithm [Widmer 96].

Maloof and Michalski implemented a variation of the time window approach [Maloof 00]. Instances older than a certain given age are deleted from the partial memory. Like the time window, the system only takes into account the last examples. However, this approach does forget observations outside the given window or older than a certain age.

Natural Selection

The natural selection approach is associated with systems implementing an ecosystem architecture of agents based on genetic algorithms. An ecosystem of specialised agents competing in parallel, gives recommendations to the user. The ecosystem evolves in the following way: agents producing the best results are reproduced with the crossover and mutation operators and others are destroyed. Amalthaea

[Moukas 97], Fab [Balabanovic 97], NewT [Sheth 93] and PSUN [Sorensen 95] use this approach.

2.4 Profile Exploitation

To recommend either products or actions to a user, a recommender system makes decisions according to the information available (items, profiles of other users on the web, etc). Thus, it is vital to select the most appropriate information with which to make decisions. Information filtering methods are based on user profile-item matching techniques and user profile matching techniques. So, regarding exploitation, three main dimensions characterise recommender systems: the information filtering method (demographic, content-based and collaborative), the item-profile matching (when content-based) and the user profile matching techniques (when collaborative). See Figure 2.2 for a general view.

2.4.1 Information Filtering Methods

There are three main information filtering methods: demographic, content-based and collaborative. Table 2.8 shows the information filtering techniques used by the various systems analysed.

Demographic Filtering

Demographic filtering approaches use descriptions of people to learn the relationship between a single item and the type of people who like it. The user profiles are created by classifying users in stereotypical descriptions [Rich 79], representing the features of classes of users. Personal data about the user is required and is used to classify users in terms of these demographic data. Classifications are used as general characterisations for the users and their interests. Commonly, the personal data is asked of the user in a registration form (see section 2.3.2). The resulting profiles span the range of information contained in the demographic database.

For instance, the method implemented by Krulwich in the LifeStyle Finder [Krulwich 97] uses a demographic system called PRIZM from Claritas Corporation which divides the population of the United States into 62 demographic clusters

NAME	METHOD
ACR News	Content-Based Filtering
Amazon	Hybrid
Amalthaea	Content-Based Filtering
Anatagonomy	Hybrid
Beehive	Collaborative Filtering
Bellcore Video Rec	Collaborative Filtering
Casmir	Hybrid
CDNow	Hybrid
Fab	Hybrid
GroupLens	Collaborative Filtering
ifWeb	Content-Based Filtering
InfoFinder	Content-Based Filtering
INFOrmer	Content-Based Filtering
Krakatoa Chronicle	Hybrid
LaboUr	Hybrid
Let's Browse	Content-Based Filtering
Letizia	Content-Based Filtering
LifeStyle Finder	Demographic Filtering
MovieLens	Hybrid
News Dude	Content-Based Filtering
NewsWeeder	Hybrid
NewT	Content-Based Filtering
Personal WebWatcher	Hybrid
PSUN	Content-Based Filtering
Re:Agent	Content-Based Filtering
Recommender	Hybrid
Ringo / FireFly	Collaborative Filtering
SIFT Netnews	Content-Based Filtering
SiteIF	Content-Based Filtering
Smart Radio	Collaborative Filtering
Syskill & Webert	Content-Based Filtering
Tapestry	Collaborative Filtering
Webmate	Content-Based Filtering
WebSail	Content-Based Filtering
WebSell	Hybrid
Websift	Hybrid
WebWatcher	Hybrid

Table 2.8: Information Filtering Method of the Systems.

according to their purchasing history, lifestyle characteristics and survey responses.

A demographic filtering system has two principal shortcomings:

- Demographic filtering is based on a generalisation of the user's interests, so the system recommends the same items to people with similar demographic profiles. As every user is different, these recommendations prove to be too general.
- The demographic approaches do not provide any individual adaptation to interest changes. The user's interests tend to shift over time [Koychev 00], so the user profile needs to adapt to change.

Nevertheless, demographic information can be a useful technique if combined with other approaches.

Content-Based Filtering

Content-based filtering approaches recommend items for the user based on the descriptions of previously evaluated items. In other words, they recommend items because they are similar to items the user has liked in the past. User profiles are created using features extracted from these items (see section 2.3.3) and each user is assumed to operate independently.

The input data most often take the form of samples of the user's interests or preferences in a given area, and the profile is a generalisation of these data which can be used generatively to carry out tasks on behalf of the user. These profiles are then used to find or recognise other items likely to be of interest. Different methods are used by the systems to match a user profile with new items and decide whether they are interesting to the user (see section 2.4.2).

In Syskill&Webert [Pazzani 96], the user rates a number of Web documents from a content domain on a binary "hot" and "cold" scale. Based on these ratings, it computes the probabilities of words being in hot or cold documents. Lieberman developed the system Letizia [Lieberman 95], which assists a user in Web browsing. Letizia tries to anticipate interesting items on the Web which are related to the user's current navigation context. For a set of links, Letizia computes a preference ranking based on a user profile. This profile is a list of weighted keywords, each one

indicating the relevance of the words found on the pages. Personalised WebWatcher [Mladenic 96] observes the individual user's choices of links on Web pages in order to recommend links on other Web pages he/she may visit later. The user does not have to provide explicit ratings. Instead, visited links are taken as positive examples, non-visited links as negative ones.

A purely content-based filtering system has several shortcomings:

- Content-based approaches are based on objective information about the items. This information is automatically extracted from various sources (e.g., Web pages) or manually introduced (e.g., product database). However, selecting one item or another is based mostly on subjective attributes of the item (e.g., a well-written document or a product with a spicy taste). Therefore, these attributes, which better influence the user's choice, are not taken into account.

- Another problem, which has been studied extensively, is over-specialisation. Content-based filtering techniques have no inherent method for generating serendipitous finds. The system recommends more of what the user has already seen and indicated a liking for. When the system can only recommend items scoring highly against a user profile, the user is restricted to seeing items similar to those already rated. In practice, additional hacks are often added to introduce some element of serendipity, in effect injecting a note of randomness.

- With the pure content-based approach, a user's own ratings are the only factor influencing future performance. However, only a few ratings are provided due to both the reluctance of users to perform actions not directed towards their immediate goals when immediate benefits are not forthcoming [Carroll 87], and the low interaction of the user with the system. Therefore, the recommendation quality is not very precise.

Nevertheless, these shortcomings can be solved by combining the content-based approach with the collaborative filtering approach (see hybrid filtering method in section 2.4.1).

Collaborative Filtering

The collaborative filtering technique matches people with similar interests and then makes recommendations on this basis. Recommendations are commonly extracted from the statistical analysis of patterns and analogies of data extracted explicitly from evaluations of items (ratings) given by different users or implicitly by monitoring the behaviour of the different users in the system. This approach is very different from content-based filtering, the other most commonly used approach. Rather than recommending items because they are similar to items a user has liked in the past, items are recommended based on other user's preferences. Rather than computing the similarity of items, the similarity among users is computed. In collaborative filtering a user's profile consists simply of the data the user has specified. This data is compared to those of other users to find overlaps in interests among users. These are then used to recommend new items. Typically, for each user a set of "nearest neighbours" is defined using the correlation between past ratings. Scores for unseen items are predicted using a combination of the scores from the nearest neighbour. This approach requires less computation than the previous one because it doesn't have to reason with the user data and it clearly leverages the commonalties between users.

Tapestry [Goldberg 92] is one of the earliest implementations of collaborative filtering based recommender systems. This system relied on the explicit opinions of people from a close-knit community, such as an office workgroup. Another popular system is GroupLens [Konstan 97], which computes correlation between readers of Usenet newsgroups by comparing their ratings of news articles. The ratings of an individual user are used to find other users with similar ratings, and their ratings are processed to predict the user's interest in new articles.

Terveen and Hill [Terveen 01] claim three essentials are needed to support collaborative filtering: many people must participate (increasing the likelihood that any one person will find other users with similar preferences), there must be an easy way to represent a user's interests in the system, and the algorithms must be able to match people with similar interests. These three elements are not that easy to develop and produce the main shortcoming of collaborative filtering systems:

- The early-rater problem: when a new item appears in the database there is no way it can be recommended to a user until more information is obtained

through another user either rating it or specifying which other items it is similar to.

- The sparsity problem: the goal of collaborative filtering systems is to help people focus on reading documents (or consuming items) of interest. As with the previous shortcoming, if the number of users is small relative to the volume of information in the system (because there is a very large or rapidly changing database) there is a danger of the coverage of ratings becoming too sparse, thinning the collection of recommendable items. Also, sparsity poses a real computational challenge as it becomes harder to find neighbours and harder to recommend items since too few people have given ratings.

- Another logic problem is that for a user whose tastes vary from the norm there will not be any other users who share his or her particular likes and dislikes, leading to poor recommendations.

- The difficulty of achieving a critical mass of participants makes collaborative filtering experiments expensive. Collaborative filtering systems require data from a large number of users before being effective as well as requiring a large amount of data from each user while limiting their recommendations to the exact items specified by those users.

- The critical dependency on the size and composition of the user population also influences a user's group of nearest neighbours. In a situation in which feedback fails to cause this group of nearest neighbours to change, expressing dislike for an item will not necessarily prevent the user from receiving similar items in the future. Furthermore, the lack of access to the content of items prevents similar users from being matched unless they have rated the exact same items.

Herlocker et al. also introduced the problem of lack of transparency in the collaborative filtering systems [Herlocker 00]. Collaborative systems today are black boxes, computerised oracles which give advice but cannot be questioned. A user is given no indicators to consult in order to decide when to trust a recommendation and when to doubt one. These problems have prevented acceptance of collaborative systems in all but low-risk content domains since they are untrustworthy for high-risk content domains.

Nevertheless, these shortcomings can be solved by combining the collaborative filtering approach with the content-based filtering approach (see hybrid approach in next section).

Hybrid Approach

Hybrid systems exploit features of content-based and collaborative filtering, since they will almost certainly prove to be complementary. On the one hand, purely collaborative systems solve the shortcomings of the purely content-based systems. The first shortcoming of content-based systems is the lack of subjective data about the items. In a collaborative system, the community of users can offer this kind of data explicitly. Subjective data can be an opinion of one item offered by a trusted friend. For instance, you can buy a spicy product because a user with similar tastes has recommended it to you. Another shortcoming of content-based systems is the lack of novelty. A perfect content-based technique would never find anything novel, limiting the range of applications for which it would be useful. Collaborative filtering techniques excel at identifying novelty using other users' recommendations and you can receive items dissimilar to those seen in the past. Finally, content-based systems lack user ratings to represent the user's interests. Collaborative systems can complete the user information with another user's experience as a basis. For instance, if you are very similar to another user and you have not rated a product, the system can use the other user's ratings to complete your interests.

On the other hand, purely content-based systems solve the shortcomings of the purely collaborative systems, the first of which is the early-rater problem. With content-based methods, new items can be recommended on the basis of their content, without the need for explicit ratings. Another advantage is that content-based systems can recommend to a user with unusual tastes without the need for a similar user, eliminating the sparsity problem of collaborative approaches. Finally, the number of participants is not important in content-based systems because they do not depend on population.

Thus, both content-based and collaborative filtering contribute to the other's effectiveness, avoiding the limitations mentioned for each system and allowing an integrated system to achieve both reliability and serendipity. Several papers have attested to the high performance of hybrid systems (e.g., [Pazzani 99] and [Good 99]).

Fab [Balabanovic 97], LaboUr [Schwab 01] and WebSell [Cunningham 01] propose a very simple method for combining the two approaches: user profiles based on content analysis are maintained and closely compared to determine users with similar preferences for collaborative recommendation.

2.4.2 User Profile - Item Matching

Typically, the user profile is used to recommend new items considered relevant to the user. Content-based filtering systems use direct comparison between the user profile and new items. Thus, a user profile-item matching technique is needed. Several techniques are studied here, whose aims are to automate the process of classifying items as relevant/not relevant, by computing comparisons between the representation of the user's interests and the representation of the items.

In the systems we analysed, the user profile - item matching techniques used are: a simple keyword matching, the cosine similarity, the nearest neighbour and typical classifiers. Table 2.9 shows the user profile - item matching techniques used by the different systems analysed.

Standard Keyword Matching

Standard keyword matching consists of a simple count of the terms which are present simultaneously in the current description of the new item and in the user profile. However, this model has some problems with the synonymy and plural meanings of some words.

An example is SiteIF [Stefani 98] which implements a standard keyword matching algorithm that consists of checking, for every word in the representation of the document, whether the context in which it occurs has been already found in previously visited documents and already stored in the semantic network.

Cosine Similarity

Cosine similarity comes from information retrieval research and is used in systems with simple user profile representation [Salton 88, Buckley 96, Yan 95, Chen 00]. An early similarity formula was used by Salton in the SMART system [Salton 83].

NAME	TECHNIQUE
ACR News	Itemset and Cluster Similarity Matching
Amalthaea	Cosine Similarity
Amazon	Unknown
Anatagonomy	Cosine Similarity
Casmir	Pre-Search Request Based Collaboration, Pot-Search Informing
CDNow	Unknown
Fab	Cosine Similarity
ifWeb	Standard Keyword Matching
InfoFinder	Boolean Search Query String
INFOrmer	Graph Comparison
Krakatoa Chronicle	Cosine Similarity
LaboUr	Bayessian Classifier, Nearest Neighbour
Let's Browse	Cosine Similarity
Letizia	Cosine Similarity
MovieLens	Cosine Similarity, Inducted Rules
News Dude	Short Term: Nearest Neighbour(Cosine Similarity) Long Term: Naive Bayesian Classifier
NewsWeeder	Cosine Similarity
NewT	Cosine Similarity
Personal WebWatcher	Naive Bayesian Classifier
PSUN	Graph Comparison
Re:Agent	Nearest Neighbour, Neural Network
Recommender	Inducted Rules
SIFT Netnews	Dot Product
SiteIF	Standard Keyword Matching
Syskill & Webert	Naive Bayesian Classifier, Nearest Neighbour, PEBLS, Cosine Similarity, Decision Tree
Webmate	Cosine Similarity
WebSail	TW2
WebSell	CBR with Nearest Neighbour(Pearson r Correlation)
Websift	Inducted Rules and Pattern Matching
WebWatcher	Cosine Similarity

Table 2.9: User Profile-Item Matching Technique of the Systems based on Content-Based Filtering.

Salton treated the index (user profile) and the search query (new item) as n-dimensional vectors (see section 2.3.1). The cosine formula calculates the cosine of the angle between the two vectors. As the cosine approaches "1", the two vectors become coincident. If the two vectors are totally unrelated, they will be orthogonal and the value of the cosine is "0". Moreover, the square of the cosine of the angle (easily computed as the normalised inner product of the two vectors) can be used to rank the items.

Nearest Neighbour

Nearest neighbour algorithms are based on computing the distance from the interested item to either the rest of the items or the classes of items in a user profile. This kind of algorithm [Duda 73] operates by storing all examples in the training set; that is, all items in the user profile. To learn the interest of an item, the algorithm assigns it to the class of the closest example. Depending on the item representation, the function to compute the distance can be a simple keyword matching or a weighted comparison [Schwab 01].

PEBLS [Cost 93] is a nearest neighbour algorithm which makes use of a modification of the value difference metric (MVDM) for computing the distance between two examples.

LaboUr [Kamba 95], News Dude [Billsus 99] and WebSell [Cunningham 01] are different examples of the nearest neighbour algorithm.

A particular case of the application of the nearest neighbour technique is Case-Based Reasoning (CBR). User profiles are represented by a collection of past experiences and, to recommend a new item, a wide amalgam of similarity measures to past items can be applied. The similarity encodes the knowledge that will assess whether an item suits the user's interests. Retrieval and adaptation techniques from CBR have become very important techniques for developing recommendation systems [Cunningham 01]. For instance, WEBSELL [Cunningham 01] applies CBR with a similarity measure based on Pearson r correlation.

Classification

Systems based on content-based filtering can handle the recommendation task as a classification task. Based on a set of item features, the system tries to induce a model for each user which allows him/her to classify unseen items into two or more classes. The typical classification categories are *interesting* and *not interesting* [Billsus 99], but the algorithm can classify items into any set of classes (e.g., relevant, undefined, not relevant). This means that the user profile is represented as a classifier: a neural network, decision tree, inducted rules or a Bayesian network (see section 2.3.1).

For instance, Re:Agent [Boone 98] implemented a neural network to divide several folders of e-mail into two categories: "work" and "other". Syskill&Webert [Pazzani 96] used a decision tree to classify Web pages into interesting/not interesting. Recommender [Basu 98] implemented a rule induction method to classify movies.

2.4.3 User Profile Matching

Systems based on collaborative filtering match people with similar interests and then make recommendations on this basis. Generally speaking the process of computing a recommendation consists of three steps: find similar users, create a neighbourhood and compute a prediction based on selected neighbours.

Find similar users

Standard similarity measures are used to compute the distance between the current user's representation and the representation of a set of users. The commonest techniques used to compute the similarity between users are nearest neighbour, clustering and classifiers. Table 2.10 shows the user profile matching techniques used by the different systems analysed.

It is important to note that, in smaller applications, the set of users, among whom similarity is being computed, may be all users; in larger systems, statistical sampling methods are used to find a representative subset for which similarity is computed. In general, systems cannot work with large sets of data containing all the users and features, since the performance of the system will gradually break down.

NAME	TECHNIQUE
Amazon	Unknown
Anatagonomy	Cosine Similarity
Beehive	Sharing news among users of the same cluster
Bellcore Video Recom	Nearest Neighbour(Pearson r Correlation)
Casmir	Pre-Search Request Based Collaboration, Pot-Search Informing
CDNow	Unknown
Fab	Cosine Similarity
GroupLens	Nearest Neighbour(Pearson r Correlation)
Krakatoa Chronicle	Cosine Similarity
LaboUr	Clustering(Nearest Neighbour - Pearson r Correlation)
MovieLens	Cosine Similarity
NewsWeeder	Cosine Similarity
Personal WebWatcher	Naive Bayesian Classifier
Recommender	Inducted Rule Execution
Ringo / FireFly	Nearest Neighbour(Mean Squared Differences, Pearson r Correlation, Constrained Pearson r Correlation, Artist-Artist)
Smart Radio	Nearest Neighbour(Pearson r Correlation)
Tapestry	Tapestry Query Language
WebSell	CBR with Nearest Neighbour(Pearson r Correlation)
Websift	Rule Execution and Pattern Matching
WebWatcher	Cosine Similarity

Table 2.10: Techniques Used by Systems based on Collaborative Filtering to Find Similar Users.

Several studies have been performed in order to reduce the data dimensionality, as for example [Hofmann 99] and [Hayes 01].

Nearest Neighbour. Nearest neighbour algorithms are applicable as a user profile-item matching technique (see section 2.4.2) as well as a method to find similar users. In the latter case, nearest neighbour algorithms are based on computing the distance between consumers based on their preference history. Predictions of how much a user will like an item are computed by taking the weighted average of the opinions of a set of nearest neighbours for that product. Nearest neighbour algorithms have the advantage of being able to incorporate the most up-to-date information rapidly, but the search for neighbours is slow in large databases. Herlocker et al. compare different nearest neighbour techniques and, as conclusions, show the results of these techniques in a specific framework and the suitability of each in different recommendation systems [Herlocker 99].

In general, two approaches are used in current systems to calculate the similarity between users: cosine similarity and correlation. Cosine similarity is applied in a way similar to the user profile-item matching technique (see section 2.4.2) and users are compared to other users by the use of two vectors.

Regarding correlation, it is easy to define similarity measures between two user

profiles working with databases of user ratings for items in which users indicate their interest in an item on a numeric scale. The typical correlation measures used in the systems analysed are the Pearson r correlation coefficient (proposed by [Shardanand 95]) and the Spearman rank correlation coefficient (proposed by [Herlocker 99]).

Another approach based on correlation between users is the entropy-based uncertainty measure. The measure of association based on entropy uses conditional probability techniques to measure the reduction in entropy of the active user's ratings which results from knowing another user's ratings. Herlocker et al. have shown that entropy has not shown itself as performing as well as Pearson r Correlation [Herlocker 99]. Shardanand and Maes, in addition to Pearson r Correlation and Constrained Pearson r Correlation, use the Mean Squared Differences algorithm, which performs well compared to the Pearson r Correlation [Shardanand 95]. Another, more complicated approach, is explained in [Greening 97].

Clustering. Some years ago, the user modeling community proposed a stereotype approach [Rich 79]. During the development stage of a system, user subgroups are identified and typical characteristics of members of these subgroups determined. During the run-time of the system, the user is assigned to one or more of these predefined user groups and their characteristics attributed to the user. The need for an empirically based pre-definition of these stereotypes is an evident disadvantage. As an alternative, the Doppelganger system used clustering mechanisms to find user groups dynamically, based on all available individual user models [Orwant 95]. Explicitly represented user models can be clustered and the descriptions of the clusters can be used like predefined stereotypes. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster.

Clustering techniques usually produce less-personal recommendations than other methods, and in some cases, the clusters have less accuracy than nearest neighbour algorithms [Breese 98]. Once the clustering is complete, however, performance can be very good, since the size of the group being analysed is much smaller.

Classification. Collaborative filtering method can be seen as a classification task [Billsus 98] as well as part of content-based filtering (see section 2.4.2). In collabo-

rative filtering, where we want to infer item interest to a user, based on similarity with other users, typically, the initial data exists in the form of a sparse matrix (see section 2.3.3), where rows correspond to users, columns correspond to items and the matrix entries are ratings. Note that "sparse" in this context means that most elements of the matrix are empty, because every user typically rates only a very small subset of all possible items. The prediction task can now be seen as filling in the missing matrix values. Since we are interested in learning personalised models for each user, we associate one classifier with every user. This model can be used to predict the missing values for one row in our matrix.

Some examples of classifiers are implemented in systems as [Basu 98], [Good 99] and [Billsus 98].

Create a neighbourhood

When systems look for similar users, they form a neighbourhood of the most similar users to the target user. Generally, two techniques have been used to determine how many neighbours to select: the correlation-thresholding technique and the best-n-neighbours technique.

The correlation-thresholding technique is to set an absolute correlation threshold, where all neighbours with an absolute correlation greater than given thresholds are selected. Setting a high threshold limits the neighbourhood to containing very good correlates, but for many users high correlates are not available, resulting in a small neighbourhood which cannot provide prediction coverage for many items.

The best-n-neighbours technique is to pick the best-fixed number of users. This technique performs reasonably well, as it does not limit prediction coverage. However, picking a larger number of users will result in too much noise for those who have high correlates. Picking a smaller number can cause poor predictions for those users who do not have any high correlates.

A different approach has been proposed in [Herlocker 99] for neighbourhood formation based on the centroid. The first step is picking the closest user to the target user and calculate the centroid. Then, other users are included in the neighbourhood based on the distance to the centroid, which is recalculated each time a new user is added. Basically, this algorithm allows the nearest neighbours to affect the formation of the neighbourhood and can be beneficial for very sparse data sets.

Computing a prediction based on selected neighbours

The final step is to derive the recommendations from the neighbourhood of users. Once the neighbourhood has been selected, the ratings from those neighbours are combined to compute a prediction, after possibly scaling the ratings to a common distribution. Different techniques are used in current systems: the most-frequent item recommendation, the association rule-based recommendation and the weighted average of ratings.

The most-frequent item recommendation looks into the neighbourhood and scans through the user's interests extracting the most frequently selected items. After all the neighbours have been accounted for, the system sorts the items according to frequency and simply returns the n most frequent items not yet selected by the active user as recommendation.

The association rule-based recommendation infers rules previously generated from the neighbourhood instead of using the entire population of users. Note that, considering only a few neighbours may not generate strong enough association rules in practice, which, in consequence, may result in insufficient items to recommend. The number of items can be augmented by using a scheme in which the rest of the items, if necessary, are computed by using the most frequent item algorithm.

Another way to combine all the neighbour's ratings into a prediction is to compute a weighted average of the ratings using the correlation as the weight. The basic weighted average makes an assumption that all users give ratings of approximately the same distribution.

The approach taken by GroupLens [Resnick 94] was to compute the average deviation of a neighbour's rating from that neighbour's mean rating, where the mean rating is taken over all items the neighbour has rated. The justification for this approach is that users may rate distributions centred on different points.

2.5 Cross-dimensional analysis

We have analysed of 37 systems following a functional approach that allows us to draw up a general taxonomy comprising 8 dimensions and based on two main groups: user profile generation and maintenance, and user profile exploitation tech-

niques. We then carried out a cross-dimensional analysis using the results of the spatial approach (see table 2.1); that is, a cross-dimension analysis among all the recommender systems of the same domain. From such an analysis, we have detected common patterns in web recommender systems, e-commerce recommender systems, item recommender systems and news recommender systems.

First, table 2.11 illustrates a cross-dimension analysis among web recommender systems. Some common features come up when we look at the different systems. First of all, we can conclude that most of the systems need feature selection and information indexing techniques to extract relevant information from text. Therefore, web recommender systems represent profiles as feature vectors, learn profiles from text through feature selection and TF-IDF techniques and match profiles with new items through cosine similarity technique. Another feature to stress is the information filtering method: web recommender systems analyse the content of web pages before recommendation, therefore implementing a content-based filtering method. Some of them, however, take advantage of collaborative techniques to improve their results.

Second, table 2.12 shows a cross-dimension analysis among the three e-commerce recommender systems analysed in this chapter. We can arrive at several relevant conclusions from this table. First, these systems do not need feature selection and information indexing techniques because the source of the information is a structured database of products. Thus, they represent the user profile as a history of *interesting/not interesting/purchased* products. Such profiles grow enormously as time passes, but e-commerce recommender systems are not interested in reducing the size of the profile because they do not want to lose information in a contraction process. So, a profile learning technique is not needed and they do not apply a profile adaptation technique to forget old interests. The large size of the user profile requires an advanced user profile - item matching technique, since the success of the recommendations depends on it. We think that this is the heart of e-commerce systems and therefore big e-commerce companies, such as Amazon and CDNow, do not publish which method they are using. Finally, it is important to note that these systems take advantage of the collaborative world, apart from the content analysis, to improve the quality of their recommendations.

Third, table 2.13 shows the cross-dimension analysis among item (e.g., movies, music,...) recommender systems. The first conclusion that we can extract from this

NAME	REPRESENT	INITIAL	LEARNING	FEEDBACK	ADAPTATION	FILTE	PROFILE MATCHING
Amalthaea	Feature Vector	Manual	F.S., TF-IDF	Ratings	Natural Selection, GFF	Content	Cos Similarity
Fab	Feature Vector	Empty	TF-IDF	Ratings	Natural Selection	Hybrid	Cos Similarity
ifWeb	Semantic Network	Training Set, Stereotyping	F.S.	Ratings	GFF	Content	Keyword Matching
Let's Browse	Feature Vector	Training Set	TF-IDF	Links, Time	Add New	Content	Cos Similarity
Letizia	Feature Vector	Empty	TF-IDF	Links, Time	Add New	Content	Cos Similarity
Personal WebWatcher	Feature Vector	Manual	TF-IDF	Links	Add New	Hybrid	Bayesian Classifier
SiteIF	Semantic Network	Empty	F.S.	Links	GFF	Content	Keyword Matching
Syskill & Webert	Feature Vector, Decision Tree	Manual, Stereo-typing	F.S., TF-IDF ID3	Ratings	Add New	Content	Bayesian Classifier, PEBLS, Cos Similarity, Decision Tree
Webmate	Feature Vector	Empty	TF-IDF	Like/ Dislike	Add New	Content	Cos Similarity
WebSail	Feature Vector	Empty	TF-IDF	Like/ Dislike	Add New	Content	TW2
Websift	Rules, Patterns	Training Set	Rule Induction	History	Add New	Hybrid	Rules, Pattern Matching
WebWatcher	Feature Vector	Manual	TF-IDF	Links	Add New	Hybrid	Cos Similarity

Table 2.11: Cross-Dimension Analysis Among Web Recommender Systems.

NAME	REPR	INIT	LEARNING	FEEDBACK	ADAPT	FILTE	MATCH
Amazon	History	Empty	Not Necessary	Ratings, History	Add New	Hybrid	Unknown
CDNow	History	Empty	Not Necessary	Ratings, History	Add New	Hybrid	Unknown
WebSell	History	Empty	Not Necessary	Unknown	Add New	Hybrid	CBR

Table 2.12: Cross-Dimension Analysis Among E-Commerce Recommender Systems.

NAME	REPRES	INITIAL	LEARNING	FEED	ADAPT	FILTE	MATCH
Bellcore Video	Ratings Matrix	Training Set	Not Necessary	Ratings	Add New	Collab	Nearest Neighbour
LifeStyle Finder	Demographic Features	Stereotyping	Not Necessary	Ratings, History	Add New	Demog	Demographic Reasoning
MovieLens	Feature Vector, Rules	Training Set	TF-IDF, Rule Learning	Ratings	Add New	Hybrid	Cos Similarity
Recommender	Rules	Training Set	Rule Learning	Ratings	Add New	Hybrid	Rules
Ringo/FireFly	Ratings Matrix	Training Set	Not Necessary	Ratings	Add New	Hybrid	Nearest Neighbour
Smart Radio	Ratings Matrix	Training Set	Not Necessary	Ratings, Implicit	Add New	Collab	Nearest Neighbour

Table 2.13: Cross-Dimension Analysis Among Item Recommender Systems.

table is that all the systems take advantage of the collaborative world to improve their recommendations. In addition, most of the systems only recommend items based on other users opinions and these systems therefore represent the user profile as a user-item ratings matrix. Using this matrix as a user profile means there is no need for a profile-learning method and recommender systems match user profiles by the nearest neighbour technique. However, it is very important that users provide relevance feedback to fulfill the matrix of the profile. Looking at the table, we can conclude that all the item recommender systems request ratings as relevance feedback. We also notice that all the item recommender systems add new information to the profile and never forget past item interests. However, we believe that a technique that will adapt the user profile as time passes, forgetting old interests, is needed in this domain.

Finally, table 2.14 shows the cross-dimension analysis among news recommender systems. Like web recommender systems, news recommender systems need feature selection and information indexing techniques to extract relevant information from text. Hence, they represent the user profile with a feature vector, or some more complex structure, such as an associative network or an n-gram. Therefore, most of them learn user profiles through feature selection and TF-IDF techniques.

NAME	REPRESENT	INITIAL	LEARN	FEED	ADAPT	FILT	MATCHING
ACRNews	Clusters	Training Set	Rule Learning, Clustering	History	Add New	Content	Cluster Matching
Anatagonomy	Feature Vector	Empty	TF-IDF	Ratings	Add New	Hybrid	Cos Similarity
Beehive	Clusters (Feature Vectors)	Empty	Clustering	History	Add New	Collab	Cluster Matching
GroupLens	Ratings Matrix	Empty	TF-IDF	Ratings, Text, Time	Add New	Collab	Nearest Neighbour
INFOrmer	Associative Network	Training Set	F.S.	Ratings	Add New	Content	Graph Comparison
Krakatoa Chronicle	Feature Vector	Empty	TF-IDF	Ratings, Implicit	Add New	Hybrid	Cos Similarity
News Dude	Feature Vector	Training Set	TF-IDF	Like/ Dislike	Short/Long Term Models	Content	Nearest Neighbour, Bayesian Classifier
NewsWeeder	Feature Vector	Training Set	TF-IDF, MDL	Ratings	Add New	Hybrid	Cos Similarity
NewT	Feature Vector	Training Set	F.S., TF-IDF	Like/ Dislike	Natural Selection	Content	Cos Similarity
PSUN	N-Grams	Training Set	F.S., N-Gram Induction	Ratings	Natural Selection	Content	Graph Comparison
SIFT Netnews	Feature Vector, Decision Tree	Training Set	TF-IDF	Like/ Dislike	Manual	Collab	Dot Product

Table 2.14: Cross-Dimension Analysis Among News Recommender Systems.

We did not detect any other pattern. We believe that this lack of common features lies in the fact that most systems have been developed following ad hoc approaches to satisfy specific application requirements. In this sense, we think that the taxonomy provided in this chapter could be a useful guide for researchers contributing to the future development of new recommender systems.

2.6 Chapter Conclusions

The unceasing growth of the Internet and its environment has brought the need for new technology to help users to find what they are looking for. The idea of recommender systems has been widely accepted among users who require assistance in searching, sorting, classifying, filtering and sharing the vast amount of information now available on the Web. This chapter has tried to gather together the state-of-the-art elements in recommender systems on the Internet.

There are several papers that have dealt with state-of-the-art recommender systems (e.g., [Sarwar 00], [Pretschner 99], [Terveen 01], [Kobsa 01]). Schafer et al., in particular, present a taxonomy of recommender systems in the e-commerce field, classifying the techniques used into three dimensions [Schafer 01]. In this chapter, we present a more complete, up-to-date taxonomy of general recommender systems on the Internet. We have analysed 37 systems, using a functional approach, and have divided our taxonomy into two main groups: user profile generation and maintenance, and user profile exploitation techniques. From this we got a basic list of 8 dimensions and we have explained, within each of these dimensions, all the techniques used in the systems we analysed. We followed that with a cross-dimensional analysis which we hope gives designers a set of clues that will help them in their work to develop new systems according to their requirements.

The remaining of our work will refer to the different dimensions and techniques analysed in order to classify the system developed and our contributions on the progress of the state of the work.

Chapter 3

CBR Approach to Recommender Systems

Recommender systems help users to identify particular items that best match their interests or preferences. In this chapter, we introduce our approach to recommendation based on Case-Based Reasoning (CBR). CBR is a paradigm for learning and reasoning through experience, based on human reasoning. We present a user model based on cases in which we try to capture both explicit interests (the user is asked for information) and implicit interests (captured from user interaction) of a user on a given item. When we apply CBR to recommender systems, some problems arise such as the adaptation of user profiles according to their interests and preferences over time or the utility problem. In order to cope with these problems, our approach includes a "forgetting mechanism" based on the drift attribute. Other systems have implemented CBR approaches to recommendation but, unfortunately, only a few evaluate and discuss their results scientifically. This chapter also proposes an evaluation technique based on a combination of real user profiles and a user simulator. The results of the simulations show that the forgetting mechanism produces an increase in precision, a decrease in recall and an important reduction of the number of cases in case bases.

3.1 Introduction

In the real world, making a selection from the incredible number of possibilities the market offers us is indeed a laborious work. Having someone as personal assistant in charge of this task would make life easier. The main function of these assistants is to advise you. In order to do this, first of all, they have to learn your tastes, interests and preferences. Then, their task consists of looking for information and analysing the market in order to find out things that may interest you. Since personal assistants are always in contact with you, they also notice your changing interests over time. If you cease to be interested in a certain thing, your personal assistant takes note and finds out what you are presently interested in.

However, hiring a personal assistant is an expense that only a very few people can afford. Recently, a great deal of effort has been put into developing electronic personal assistants. In an attempt to model the behaviour of a real personal assistant, AI research has focussed on recommender systems. Recommender systems draw on previous results from machine learning and other AI technology advances. Among the various machine-learning technologies, we concentrate on Case-Based Reasoning (CBR) as a paradigm for learning and reasoning through experience, as personal assistants do. The main idea of CBR is to solve new problems by adapting the solutions given for old ones.

As noted in sections 2.3.1, 2.3.3 and 2.4.2, when we apply CBR to recommender systems, the importance of the recommendation process lays with the case base representation. We propose, as a representation, a list of experiences (cases) of the user in certain items. Experiences are represented by means of objective attributes describing the item (case definition) and subjective attributes describing implicit or explicit interests of the user in this item (case solution). Assuming that the user's interest in a new item is similar to the user's interest in similar past experiences, when a new item comes up, the recommender system predicts the user's interest in the new item based on interest attributes of similar experiences.

However, when we apply CBR to recommender systems, there are two things missing:

- On the one hand, humans have a vast store of experience on which to base their decisions. When a new problem comes up, humans look for similar prob-

lems and try to solve it based on the most similar experiences. However, the time dimension is also present in the human reasoning process. It means that humans have in mind the most recent cases and give them the greater importance when making a decision. When we are dealing with human interests and preferences, the relevance of the most recent cases becomes even more important. Human interests evolve as time passes and what humans like in the present is more important than what humans liked in the past. In CBR, all the cases in the case base have the same relevance when they are retrieved. Therefore, if CBR is to be based on human reasoning, the relative relevance of cases according to time should be taken into account.

- On the other hand, it should be noted that presumably, with a larger set of cases, the system gives better results as long as the cases cover a wide range of problems. However, several authors claim that when the case base reaches a critical number of cases, the performance of the system does not improve but often gets worse [Leake 98]. Thus, one of the main drawbacks of CBR is the *utility problem*: the uncontrolled growth of case bases may degrade system performance as a direct consequence of the increased cost of accessing memory.

Therefore, there is a need for a technique that controls the relevance of cases and forgets irrelevant ones. In order to meet this need, we propose the *drift attribute*. The main function of the drift attribute is to conserve relevant cases in the case base while forgetting irrelevant ones, thus keeping the case base to a reasonable size and solving the utility problem.

With respect to relevance, the drift attribute allows irrelevant cases that do not provide good recommendations to be forgotten, adapting the case base to the changing interests of the user. A relevant case is a case that leads the recommendation algorithm to successful recommendations. Thanks to the drift attribute, we implement a "forgetting algorithm" that distinguishes which interests are outdated and which are up-to-date. In order to know which interests are the most up-to-date, all new cases have to be retained. Then, by analysing the entire case base, the forgetting algorithm decides which cases to retain and which to delete.

With respect to the utility problem, the drift attribute, apart from adapting the case base over time, forgets (deletes) irrelevant cases from the case base, thus maintaining a manageable size and solving the utility problem.

Other systems have implemented forgetting mechanisms but, unfortunately, only a few evaluate and discuss their results scientifically. This is in part due to the fact that, to date, it is very difficult to determine how well personalisation systems work, since this involves purely subjective assessments. This chapter also proposes an evaluation technique based on a combination of real user profiles and a user simulator that we call *profile discovering*. Thanks to the profile discovering technique, we are able to perform various evaluation measures regarding the forgetting mechanism proposed being able to demonstrate its behaviour experimentally. Moreover, the different parameters used in the system can be analysed on an experimental basis of repeatable results. The results of the simulations show that the forgetting mechanism produces an increase in precision, a decrease in recall and an important reduction of the number of cases in the case base for all the recommendation algorithms tested, while preserving fallout and accuracy.

The outline of this chapter is as follows: the next section contextualises our proposal within the current state-of-the-art. Then, our approach to applying CBR to the recommendation domain is introduced in section 3.3, and the particularities of the "drift attribute" are detailed in section 3.4. How we evaluate this forgetting mechanism and the experimental results obtained are explained in section 3.5 and section 3.6 respectively. Finally, section 3.7 concludes this article.

3.2 Related Work

A few research groups are investigating the application of CBR concepts and techniques to recommendation. Cunningham et al. apply retrieval and adaptation techniques from CBR to product recommendation systems, particularly to their WebSell system [Cunningham 01]. Websell represents the user profile by means of two product selection lists that are used for collaborative recommendation service. These selection lists contain both interesting and uninteresting products which, as in our approach, are used as a case base to proactively recommend new products to the user. Besides the lists, user profiles store more information (e.g., personal information, domain preferences), which can be compared to our approach (explicit interests). The main difference is that we explicitly keep a set of interest attributes as a solution for each case instead of general domain preferences deduced from the lists as WebSell does.

All "FindMe" systems [Burke 97] implement a similar CBR. They retrieve items that meet certain constraints and rank the results according to programmed criteria. For instance, the restaurant recommender Entree [Burke 00] makes its recommendations by finding restaurants in a new city similar to restaurants the user knows and likes. The main difference with our system is that Entree does not use an interest extensive representation of items in the case base as we do.

Adaptive Place Advisor [Goker 00] introduces an innovative approach to the retrieval phase based on diversity, which controls the repetition of the same recommendation to a user. Thus, they avoid recommending the same restaurant in a short period of time to the user. In our view, this approach, although innovative, could lead to somewhat suspicious behaviour of a system which, with the same inputs, gives different outputs in successive moments, one of which must be untruthful or untrustworthy.

The main novelty of our approach compared to all of these previous recommender systems based on CBR is the inclusion of the drift attribute that adapts the interests of the user over time and controls the case base size. Using the drift attribute is a new idea we have introduced into CBR recommendation systems, thus, there is no previous work directly related to this concept. However, this idea has been implemented in many other CBR applications and, more generally, in instance-based learning (IBL) applications, and in a few non-CBR recommender systems.

In the context of CBR/IBL systems, a great deal of work has gone into the problem of controlling the size of case bases [Wilson 00]. Several reduction techniques have been proposed, most of which can be classified in two main groups:

- Decremental techniques. These consist of beginning with a full case base containing all the cases and progressively removing the irrelevant ones in order to obtain the optimum size (e.g., RNN [Gates 72], ENN [Wilson 72], k-NN [Tomek 76], DROP1-DROP5 [Wilson 00] or DEL [Wilson 00]). These methods are not useful for recommender systems since the initial case bases are empty or contain only a few initial cases.

- Incremental techniques. These start with an empty case base and new cases are stored selectively. The criteria of selection is mainly what distinguishes one technique from another. For example, there are techniques based on the Nearest Neighbour, such as CNN [Hart 99] or SNN [Ritter 75], which only

retain cases when the existing ones in the case base lead to a classification error. The main shortcoming of these approaches is that they do not take into account the relevance of cases when deciding whether to retain. In contrast, some reduction mechanisms, such as IB3 [Aha 91, Aha 92] or MCS [Brodley 93], assign indexes to the cases or even to the individual attributes of cases, which control the relevance of each case. Only cases with an adequate relevance are retained. Every time a prediction is performed, these indexes are adjusted according to the success of the result. These latest methods are perfectly suitable for dynamic environments such as recommender systems. However, they do not take into account change in user interests over time. One important drawback of these methods is that if a new case does not contribute to the case base, it is not retained. In recommender systems, retaining the latest cases is vitally important, since they denote the current interests of the user.

A very different kind of reduction technique uses, for example, algorithms that generalise cases into rules which are then used to decide which cases to retain [Chang 74]. Other algorithms generate clusters or prototypes of cases representing classes wherein cases can be classified [Domingos 95, Domingos 96]. Both methods have drawbacks when dealing with dynamic systems and especially with recommender systems whose case bases have to be adapted to the user preferences over time. Zhang and Yang maintain feature weighting in a dynamic context through an integration with a learning system inspired by a back-propagation neural network [Zhang 98]. This method can suffer from the typical neural network problem of divergence. In a domain constrained by adaptation to new interests, the divergence problem could occur too frequently, leading the system to a fatal performance.

In the context of recommender systems, maintenance is rendered even more difficult since the relevance of cases depends on the user preferences and the necessity of handling change in human interests over time. Mitchell et al. proposed learning the description of the user's interests from only the latest observations, with a time window [Mitchell 94]. Maloof and Michalski suggested giving examples an age and deleting instances from the partial memory older than a certain age [Maloof 00]. Billsus and Pazzani implemented a dual user model consisting of both a short-term and a long-term model of user's interests [Billsus 99]. Finally, Webb and Kuzmycz introduced forgetting old interests with the gradual forgetting function [Webb 96]. The main idea behind this is that natural forgetting is a gradual process.

Recommender systems such as SiteIF [Stefani 98] or LaboUr [Schwab 01] implement this proposal.

All the forgetting mechanisms proposed in current state-of-the-art recommender systems simply forget old cases without taking into account the relevance of such cases. It means that they do not take into account which cases provide the most successful recommendations. What we propose is a forgetting mechanism that, of course, forgets outdated cases, but in addition, based on the CBR reduction techniques, gives each case a weighting value depending on their relevance. The most relevant cases are retained and used in order to make new recommendations while the irrelevant ones are deleted.

3.3 Case-Based Recommendation Framework

The core of CBR is a case base which includes all the previous experiences that can give us information we can use to deal with new problems. Then, through the similarity concept, the most similar experiences are retrieved. However, similarity is not a simple or uniform concept. Similarity is a subjective term that depends on what one's goals are. For instance, two products with the same price would get maximum similarity if the user was interested in products with that same price, but would get very different similarity for other concepts, such as quality or trademark.

In our approach, the case base represents the user profile and consists of a set of previous experiences (cases); that is, items explicitly and/or implicitly assessed by the user. Each case contains the item description (attributes describing a restaurant in the example) and the interest attributes describing the interests of the user concerning the item. These latter attributes can be explicitly given by the user or implicitly captured by the system.

Assuming that the user's interest in a new item is similar to the user's interest in similar items, in order to evaluate whether a new item could interest the user, the recommender system searches the case base for similar items. If the interest the user showed in them is high enough, the new item is recommended to the user. This kind of recommendation based on similar items is our approach to content-based filtering (see section 2.4.1).

With regard to the CBR cycle, we reassess the different phases as follows:

1. In the retrieval phase, i.e. a new item, the system searches for similar items in the case base in order to find out whether the user might be interested in them. Local similarity measures are based on item attributes.

2. In the reuse phase, i.e. the retrieved set of similar items, the system calculates a confidence value of interest to recommend the new item to the user based on explicit and implicit interests and the validity of the case according to the user's current interests.

3. In the revision phase, i.e. the relevance feedback of the user, the system evaluates the user's interest in the new item. The idea is to track user interaction with the system to get to know relevant information about the user's interest in the recommended item, as well as explicit and implicit information, in order to retain the new case.

4. In the retain phase, the new item is inserted in the case base with the interest attributes that were added in the revision phase. In order to control the case base size, it is also important to know if the user ever gives new feedback about items in the case base. In such a case, it is necessary to forget these interests with time. We propose the use of a new attribute that we call the *drift attribute*, which will be aware of such changes in user preferences and contribute to case maintenance.

In the following sections the structure of the case base and the different CBR phases of the new approach are detailed.

3.3.1 The Case Base

A case-based reasoner is heavily dependent on the structure / representation and content of its collection of cases. In our approach, a case represents the user's experience concerning a certain item. Cases are split into two parts: the first, a set of objective attributes describing the item (the definition of the problem in CBR terminology) and the second, a set of interest attributes describing the user's interest in the given item (the solution to the problem in CBR terminology). Thus, given a set of items

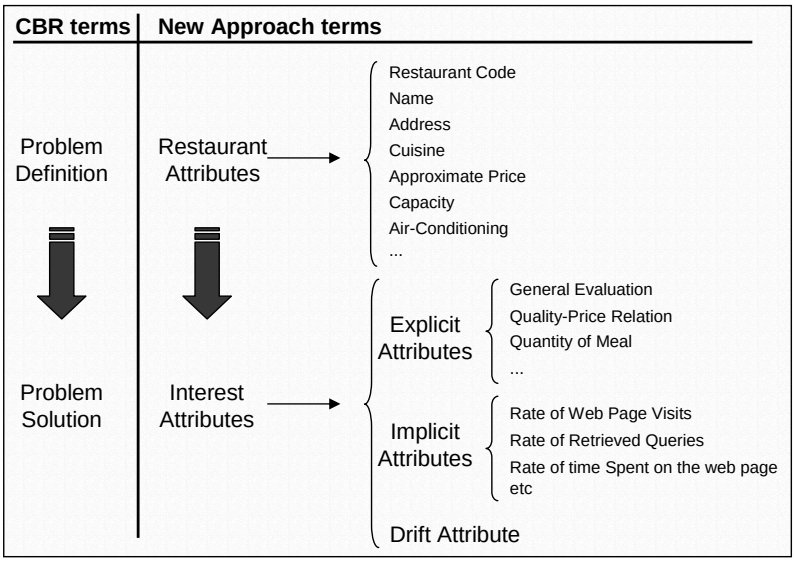


Figure 3.1: An Example of Case Representation in the Restaurants Domain.

$$P = \{p_1, p_2, \ldots, p_s\},$$

each item is characterised by a set of objective attributes,

$$p_i = \{at_{i_1}, at_{i_2}, \ldots, at_{i_n}\}$$

being At, the set of all possible attributes. In general, objective attributes do not tend to be very complex, consisting largely of descriptive adjectives, nouns or values. For example, when the CBR goal is to recommend restaurants (see Figure 3.1), the system can deal with features of capacity (e.g., "100 places" or "150 places"), qualities of the cuisine (e.g., "traditional", "creative" or "bland") or approximate price (e.g., "from $10 to $15" or "from $20 to $30").

Each user has a different degree of interest in any given items. As seen in section 2.3.4, such interest can either be expressed by the user (explicit attributes) or captured automatically by the system as a result of user interactivity (implicit attributes). Explicit interest provides more confidence in the recommendation process. However, this is not always available. Implicit interest is useful when deciding upon interesting items for the user. In our model we distinguish both kinds of user

interactions: explicit from implicit and, therefore, a hybrid approach. We name the set of explicit interest as

$$Int^e = \{int_1^e, int_2^e, \ldots, int_m^e\}$$

and the set of implicit interest as

$$Int^i = \{int_1^i, int_2^i, \ldots, int_l^i\}$$

Both int_j^e and int_j^i are defined in [0,1].

Each user has experiences in several items. An experience keeps information about the objective attributes of a given item as well as subjective information regarding the interest of the user in that item. Thus,

$$E_i =< p_i, Int_i^e, Int_i^i, \delta_i >,$$

where $p_i \subset P$ is the set of objective attributes of the item, $Int_i^e \subset Int^e$ is the set of explicit interest, $Int_i^i \subset Int^i$ is the set of implicit interest, and δ_i is a temporal parameter in [0-1] that indicates the relevance of the experience. This parameter is called the drift attribute (see section 3.4). Initially δ is set to 1, and is updated according to the evolution of the user profile.

Finally, if a case represents the experience of the user inn a given item, the complete case base constitutes the user profile representation which models the user. So, the recommender system keeps a case base for each user representing their profile.

In order to start recommending to the user, the system needs to fill in the user profile; that is, the set of initial experiences in the case base. The initial experiences are generated through the use of the training set technique (see section 2.3.2). That is, users are prompted to a set of items and they have to fill in information regarding their interest in these items. The item set consists of a collection of selected items. For each item in the set, the system asks the user about the explicit interest and also

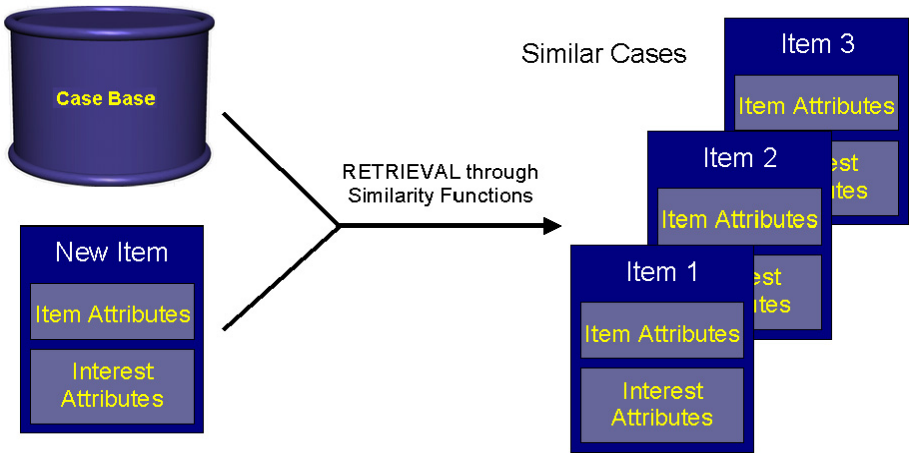


Figure 3.2: Retrieve Phase.

gathers information related to implicit interests. The selection of a suitable initial item set is indeed very difficult. Users are often asked about items that they do not know and they have to invent an evaluation taking into account their attributes. Despite such shortcomings, we have chosen this technique because of its simplicity. Other advantages and disadvantages of this kind of initial experiences generation technique have been broadly discussed in section 2.3.2.

3.3.2 The Retrieval Phase

In CBR terminology, the retrieval task starts with a new problem description and ends when the best matching set of previous cases has been found. When we apply CBR to recommendation, this phase has the same purpose, but instead of retrieving similar problems, the system retrieves similar items. Thus, the retrieval task ends when the set of best matching previous items has been found (see Figure 3.2).

The most important step in the retrieval phase of CBR is to define the degree of similarity between cases. The success of CBR systems depends primarily on the capacity of the system to exhibit how similar two cases are. With an efficient similarity measure, given a case we can obtain an ordered list of similar cases. Taking advantage of this concept, when a user likes an item, the recommender system can recommend to him/her a list of similar items that the user should like.

The degree of similarity between two items is computed by a global similarity function. The global similarity is calculated from a weighted ponderation of the

various attribute similarities. For the numeric attributes, linear and exponential functions have been designed following [Falkman 00, Wilson 97]. For labeled attributes, we have generated similarity tables, where the similarities among the different attribute labels are predetermined. For more details on similarity functions see [Vilà 02].

Once the similarities between the new case and the cases in the case base are calculated, a set of best matches is chosen. In our implementation, we select the x best cases provided which exceed a minimum selection threshold.

3.3.3 The Reuse Phase

The reuse phase consists of adapting the old solutions of the retrieved cases to the new problem based on the differences among them. Once the system has retrieved a set of previous items (the most similar ones), the system knows the user's interest in similar items through the interest attributes (solution in CBR terminology). Assuming that the user's interest in a new item is similar to the user's interest in similar items, in the reuse phase, the recommender system calculates an interest confidence value for the new item. This value is used to decide whether to recommend the new item to the user.

The interest confidence value is a composite of the item interest values of the similar items selected in the retrieve phase (see Figure 3.3). We calculate this in a two-step process.

First, the item interest value V of each case is computed based on its interest attributes as follows:

$$V_p = \delta_p * g(f^e(Int_j^e), f^i(Int_j^i)) \tag{3.1}$$

where f^e is the function that combines the explicit interest, f^i is the function that combines the implicit attributes, g is the function that combines the results of f^e and f^i, and finally δ_p is the drift attribute.

On the one hand, the idea of the function f^e is to aggregate the explicit attributes in order to obtain a general explicit evaluation of the user concerning the given item. We have implemented this function as a weighted arithmetic average (WA) where

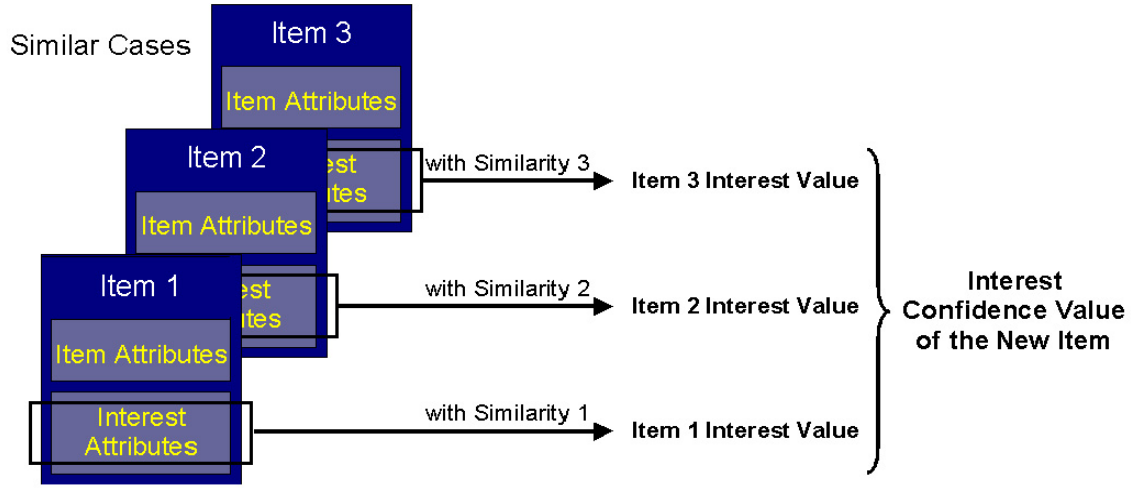


Figure 3.3: Reuse Phase.

each attribute has a weight assigned. The function is defined as follows:

$$f^e = \sum_{j=1}^{|p_i^e|} w_j * Int_j^e \tag{3.2}$$

where $|p_i^e|$ is the number of explicit attributes of the item p_i and w_j are the weights assigned to the different explicit attributes. We think these weights are very subjective and, therefore, the user has to set them up. For this reason, at a preliminary phase, recommender systems ask the users for the relevance they think the different explicit attributes have. In the previous example, the user would have to rank what they think is more important in a restaurant (quality/price relation, quantity of meal, ...) with the magnitude of such importance.

On the other hand, f^i aggregates the implicit attributes. We think this function is highly dependent on the user's behaviour when using a recommender system. For this reason, we believe an expert should decide which information to capture implicitly and how to aggregate it.

Finally, function g aggregates implicit and explicit general evaluations. We have implemented this function as WA that gives more importance to explicit attributes (objective ones) than to implicit ones (subjective):

$$g(e, i) = \rho_e * e + \rho_i * i \tag{3.3}$$

Second, the interest confidence value I of a new item r is a weighted ponderation function of the item interest value of each similar item:

$$I_r = \frac{\sum_{i=1}^{x}(Sim(r,i) * V_i)}{\sum_{i=1}^{x} Sim(r,i)} \tag{3.4}$$

where x is the number of similar items, $Sim(r,i)$ is the similarity between item r and item i (computed in the retrieval phase) and V_i is the item interest value of item i. In this way, the most similar items are the most relevant in the final result.

Finally, if the interest confidence value of the new item is greater than a certain value (a confidence threshold τ^+), the item is recommended to the user. Otherwise, the system ignores it, the CBR cycle finalises and there is no recommendation to the user. The item is not interesting enough to the user so the system should not bother him/her with it.

3.3.4 The Revision Phase

Typically, the revision phase consists of evaluating the case solution generated by the reuse phase and learning about it. If the result is successful, then the system learns from the success (case retainment), otherwise it is necessary to repair the case solution using domain-specific knowledge. With regard to our approach, in the revision phase, as in the case of relevance feedback from the user, the system is able to evaluate the user's interest in the recommended item. The idea is to track user interaction by filling in the interest attributes of the item (case).

As shown in Figure 3.1, the interest attributes are distributed in two main groups: implicit and explicit attributes. Obviously, implicit attributes come from implicit feedback from the user, and explicit attributes come from explicit feedback. The idea is to find out the user's interest based on a hybrid relevance feedback system (see section 2.3.4). The user is explicitly asked about the new item but, taking into account that users are very reluctant to give explicit feedback [Carroll 87], the system tracks the user interaction with the system and tries to include additional information.

In CBR systems the solution is successful or wrong. When the solution is successful, the system retains the case, inserting it into the case base. But when the

solution fails, the system is also interested in retaining the reason for the failure thus, there is an investigation task to find out additional information about the case. In the recommendation field, the user's interest can also be positive or negative but, contrarily to the previous situation, the system is interested in retaining both positive and negative feedback. It is equally important to keep positive and negative information about the interests of the user, since it is useful to know what the user really "loves" and "hates". Thus, in this approach, in contrast to what usually happens in CBR systems, there is no investigation task to find out why the user is not interested in the new item, the relevance feedback information is captured and kept in the Int^e and Int^i sets, included in each experience of the user profile. Therefore, in avoiding the investigation task, typically accomplished by a human expert, we get a completely automatic system.

3.3.5 The Retain Phase

In our approach, the new item is inserted into the case base with the interest attributes that were added in the revision phase. However, if the user did not give either explicit or implicit feedback, the user has not any experience on it and, therefore, the case is not introduced into the case base.

Moreover, when the user gives explicit or implicit feedback about an existing item in the case base, the case is updated. For example, if the user consults the web page of an item, the interest attribute representing the number of visits to the web page and the attribute representing the time spent looking at the web page are increased.

In order to control the case base size, it is also important to know whether users ever give new feedback about items in the case base. If not, it is necessary to forget their interests with time. This problem is solved with the drift attribute.

3.4 The Drift Attribute

The drift attribute is one of the interest attributes (see Figure 3.1) and its function is to forget the cases in the case base according to their age and their relevance. There is one drift attribute per case and a forgetting algorithm updates it according

to the user-system interaction.

The forgetting algorithm, based on the drift attribute, works as follows:

- The drift attribute value is confined to the [0-1] interval.
- New items are inserted in the case base with the maximum drift value when the user shows some interest in them.
- The value of the drift attribute is decreased over time, emulating the gradual process of people losing interest in something. The decreasing function is a simple one where the drift attribute δ_q of a case q is decreased by multiplying the last drift value by a factor β of between 0 and 1.

$$\delta_q = \delta_q * \beta \tag{3.5}$$

  The key issue then, is when to apply the decreasing function. We have to take into account that some users interact with the system more frequently than others. Therefore, the decreasing function should depend on the user interaction, rather than on a certain number of days or weeks. Our system decreases drift attributes each time a new item is incorporated into the case base.

- The value of the drift attribute is increased (rewarded) if the retrieved case results in a successful recommendation. The rewarding function is as simple as the decreasing one. The drift attribute δ_q of a case q is increased by multiplying the last drift value by a factor λ greater than 1 and ensuring that the final value is at most 1.

$$\delta_q = \delta_q * \lambda \tag{3.6}$$

- Finally, the value of the drift attribute is decreased (penalised) if the retrieved case results in a failed recommendation. The penalising function decreases the attribute δ_q of a case q by multiplying the last drift value by a factor σ smaller than 1 and ensuring that the final value is at least 0.

$$\delta_q = \delta_q * \sigma \tag{3.7}$$

When a case reaches a drift value under a certain threshold (ξ), it is discarded. If the drift value is low enough, it does not make sense to retain the item in the case base. The confidence value for the interest that this item gives is insignificant and it is a useless case that only contributes to increasing the size of the case base and decreasing the performance of the system. Therefore, removing cases with a low drift value is the best solution for automatically controlling the size of the case base.

The forgetting mechanism based on the drift attribute needs a setup phase where the parameters are optimised to get the best performance out of the system. Different results can be obtained by changing the rewarding factor (λ), the penalising factor (σ), the decreasing factor (β) and the threshold (ξ). Finding out the optimal values is an empirical task based on different measures to evaluate the overall system performance. From our experiments, we have chosen $\lambda = 1.05$, $\sigma = 0.95$, $\beta = 0.98$ and $\xi = 0.7$ as the parameter values that give the best system performance, as shown in the following sections.

The main difference between other gradual forgetting approaches and ours, is the fact that we emphasise relevant information instead of simply an age. Moreover, for example in [Webb 96, Schwab 01], they have an age for all the items of a given topic and when some event affects one of the items in the topic this age is modified in such a way as to affect all the items in that topic. We think that this reduces system performance, especially when the same topic includes a large set of items. Because, if in the same interest topic there is one single item that the user is interested in, the topic never drifts, even if the user is not interested in all the other items in the set. Alternatively, in our approach we assign a weight to each item; thus, each one is treated individually, hence solving this problem.

3.5 Evaluation Methods

In this section we describe the measures used to evaluate our proposal and how the results have been acquired.

3.5.1 Evaluation Metrics

A set of metrics are proposed in order to evaluate recommender systems: precision, recall, f-measure, fallout, ncases, diversity and accuracy.

Precision

The Precision measure [Salton 83] is the fraction of the selected items which are relevant to the user's information need. It is also a measure of selection effectiveness and represents the probability that a selected item is relevant. Precision is calculated with the following formula:

$$P = \frac{s}{n} \tag{3.8}$$

where s is the number of successful recommendations and n is the number of recommendations. The result is a real value ranging from 0 to 1. Precision can also be seen as the probability that a recommendation be successful.

Recall

The Recall measure [Salton 83] is the fraction of the actual set of relevant items which have been correctly classified as relevant. It is a measure of selection effectiveness and represents the probability that a relevant document will be selected. It is interesting to evaluate the number of recommendations that the system makes, since, of course, a recommendation algorithm that recommends all the items will obtain all the possible successes. Recall is computed as follows:

$$R = \frac{n}{t} \tag{3.9}$$

where n is the number of recommendations and t is the total number of possible

recommendations. The result of this formula is a real number ranging from 0 to 1. Recall can also be seen as the probability that an item be recommended.

F-Measure

It is, on occasion, important to evaluate precision and recall in conjunction, because it is easy to optimise either one separately. The F-Measure [Lewis 94] consists of a weighted combination of precision and recall which produces scores ranging from 0 to 1. When recall increases, precision decreases. Van Rijsbergen proposed a balanced weighting measure between precision and recall called the f-measure [van Rijsbergen 79]. However, we have used a variation of this measure, where the weights are controlled by a parameter b [Lewis 94]. This new approach is calculated as follows:

$$FM = \frac{(b^2 + 1) * P * R}{b^2 * P + R} \tag{3.10}$$

where P is precision, R is recall and b is the weighting factor. For example, $b = 0.0$ means that $FM = precision$; $b = unlimit$ means that $FM = recall$; $b = 1.0$ means that recall and precision are equally weighted; $b = 0.5$ means that recall is half as important as precision; and $b = 2$ means that recall is twice as important as precision. We can also see this measure as a modification of precision by recall or viceversa.

Fallout

The Fallout measure [Salton 83] is the fraction of the non-relevant items selected. It is a measure of rejection effectiveness. We use Fallout to evaluate the percentage of failed recommendations. It is computed like precision, but instead of measuring the recommendations successfully evaluated by the user, we take into account the number of recommendations that the user has evaluated as bad. Fallout is calculated with the following formula:

$$F = \frac{u}{n} \tag{3.11}$$

where u is the number of failed recommendations and n is the number of recom-

mendations. Fallout can also be seen as the probability that a recommendation be a failure. The result is a real value confined to the [0-1] interval, although fallout charts represent F normalised between 0 and 100. A fallout value close to 0 means that the system never recommends bad choices; a fallout value of 1 means that the system is always recommending uninteresting items to the user.

NCases

The study of the average number of items (cases) contained in the user profile (case base) over time is very important, since it is desirable to reduce the size of the user profiles (solving the utility problem) while preserving or even increasing precision (while adapting the profile to the user). Certainly, the forgetting mechanism will reduce the time and the capacity needed by the algorithms to perform a recommendation. Thus, NCases is calculated as follows:

$$NC = \frac{\sum_{i=0}^{k} |NC_i|}{k} \tag{3.12}$$

where NC_i is the number of items at the moment i, and k is the number of moments. That is, the simulation time has been split into k units and, in each unit, the number of cases in the case base NC_i has been measured. At the end of the simulation, the average is computed. NCases is not normalised, therefore, this number is relative to the total number of possible recommendations. What we want to study is the difference between the different NCases from the point of view of different parameters that the forgetting mechanism depends on.

Diversity

How the reduction of the number of items contained in the user profile affects the diversity within the resulting profiles is an interesting phenomenon for study. To evaluate the diversity, we propose using a well-known clustering method that calculates the number of groups of similar items contained in the profile. The clustering algorithm that we have implemented belongs to a particular subset of clustering methods knows as SAHN [Sneath 73]: Sequential, Agglomerative, Hierarchical and Non-overlapping methods. The proposed algorithm can be summarised as follows:

- STEP 0: Construction of an initial similarity matrix that contains the pairwise measures of proximity between the different items of the user profile.
- STEP 1: Selection of the two items that are most similar. These alternatives will form a new cluster.
- STEP 2: Modification of the similarity matrix creating a cluster with the selected items and recalculating the similarity between the new cluster and the remaining objects. Similarity is calculated with an Arithmetic Average criterion where the similarity between a given item and the cluster is the average similarity between the items composing the cluster and the given item.
- STEP 3: Repeat steps 1-2 until the two most similar items have a similarity value over a threshold α. This threshold has to be defined previously, taking into account that it determines the abstraction level achieved. Increasing the threshold we obtain a smaller number of wider (more general) clusters.

The number of clusters obtained after the execution of the proposed algorithm is the diversity measure that allows system simulations performed with different parameters to be compared.

Thus, a key task is to select a suitable α. Depending on this parameter, the number of clusters constituting the user profile will change. A low α means that only the most similar cases join up and, therefore, the algorithm gives a high number of clusters. Contrarily, a high α results in a low number of clusters since only the most different ones do not join up.

Accuracy

Accuracy is the most frequently used metric for evaluating systems. Typically, it is defined as the percentage of correctly classified items. For instance, the number of interesting news articles divided into the total number of news articles in a newspaper. However, Sarwar et al. gather and classify different ways to measure it [Sarwar 98] from prior research:

- Statistical Recommendation Accuracy: measures the closeness between the numerical recommendations provided by the system and the numerical ratings entered by the user for the same items. Three versions of this measure are used:

  - Correlation is a statistical measure of agreement between two vectors of data, usually between ratings and predictions. The Pearson Correlation r Coefficient is the most commonly used. A higher correlation value indicates more accurate recommendations.
  - The mean absolute error (MAE) is a measure of the deviation of recommendations from their true user-specified values. The lower the MAE, the more accurately the recommendation engine predicts user ratings.
  - The root mean squared error (RMSE) is a measure of error biased to weigh large errors disproportionately heavier than small errors. A low RMSE indicates better accuracy.

- Decision-Support Accuracy: measures how effectively recommendations help a user select high-quality items. Three versions of this measure are used:

  - The reversal rate is a measure of how often the system makes big mistakes which might undermine the confidence a user has in the recommendation systems. Low reversals refer to cases in which the user strongly dislikes an item the system has strongly recommended. High reversals are cases in which the user strongly likes an item, but the system gives a poor recommendation for it.
  - The ROC sensitivity is a measure of the diagnostic power of a filtering system. Operationally, it is the area under the receiver operating characteristic (ROC) curve; a curve which plots the sensitivity and specificity of the test. Sensitivity refers to the probability of a randomly selected good item being accepted by the filter. Specificity is the probability of a randomly selected bad item being rejected by the filter. Therefore, the ROC sensitivity measure is an indication of how effectively the system can steer people towards high-rated items and away from low-rated ones.
  - The PRC sensitivity is a measure of the degree to which the system presents relevant information. Operationally, it is the area under the precision-recall curve (PRC). Precision measures the percentage of selected documents which are relevant; recall measures the percentage of relevant documents selected. Hence, precision indicates how selective the system is and recall indicates how thorough it is in finding valuable information. A higher value is more accurate.

In particular, we want to analyse a statistical recommendation accuracy, since the result that we want to obtain is the correlation between the predictions of the recommender algorithm and the real evaluations of the user. The formula used to calculate the accuracy is the following:

$$A = \frac{\sum_{i=0}^{m} |Pred_i - Real_i|}{m} \tag{3.13}$$

where $Pred_i$ is the prediction of the item i, $Real_i$ is the real evaluation of the item i and m is the number of test examples. The result is a value ranging from 0 to 1, although the charts are illustrated with a scale from 0 to 100.

3.5.2 Results Acquisition

Acquiring results that can be used to compute evaluation measures is a critical task in the evaluation of recommender systems. Below, we describe some techniques that have been used in the current state-of-the-art. Then, we present our proposal for evaluating recommender systems, namely what we have called "profile discovering" and an extension of this proposal for performing cross-validations.

Related Work

In the current state-of-the-art, recommender systems use one of the following approaches in order to acquire the results for evaluating the performance of their systems: a real environment, an evaluation environment, the logs of the system or a user simulator.

First, results obtained in a real environment with real users is the best way to evaluate a recommender system. Unfortunately, only a few commercial systems like Amazon.com [Amazon 03] or CDNow.com [CDNow 03] can show real results based on their economic effect.

Second, evaluation environments are an alternative for some systems to be evaluated in the laboratory by letting a set of users interact with the system over a period of time. Usually, the results are not reliable enough because the users know the system or the purpose of the evaluation. An original approach was accomplished by NewT [Sheth 94]; in addition to the numerical data collected in the evaluation

sessions, a questionnaire was also distributed to the users to get feedback on the subjective aspects of the system. The main problem of the real and the evaluation environments is that repetition of the experiments, in order to evaluate different algorithms and parameters, is impossible.

Third, the analysis or validation of the logs obtained in a real or evaluation environment with real users is a common technique used to evaluate recommender systems. A frequently used technique is the "10-fold cross-validation technique" [Mladenic 96]. It consists of predicting the relevance (e.g., ratings) of examples recorded in the logs and, then, comparing them with the real evaluations. These experiments are perfectly repeatable, provided that the tested parameters do not affect the evolution of the user profile and the recommendation process. For example, the log being validated would be very different if another recommendation algorithm or another forgetting mechanism had been tested. Therefore, since the majority of the parameters condition the recommendation process over time, generally, experiments cannot be repeated.

Finally, a few systems are evaluated through simulated users. Important issues such as learning rates and variability in learning behaviour across heterogeneous populations can be investigated using large collections of simulated users whose design was tailored to explore those issues. This enables large-scale experiments to be carried out quickly and also guarantees that experiments are repeatable and perfectly controlled. It also allows researchers to focus on and study the behaviour of each sub-component of the system, which would otherwise be impossible in an unconstrained environment. For instance, Holte and Yan conducted experiments using an automated user called Rover rather than human users [Holte 96]. NewT [Sheth 93] and Casmir [Berney 99] also used a user simulator to evaluate the performance of systems. The main shortcoming of this technique is that, at present, it is impossible to simulate the real behaviour of a user. Users are far too complicated to predict, at every moment, their feelings, their emotions, their moods, their anxieties and, therefore, their actions.

"Profile Discovering"

In order to solve all the shortcomings of the current techniques while benefitting from their advantages, we propose a method of results acquisition called *"the profile*

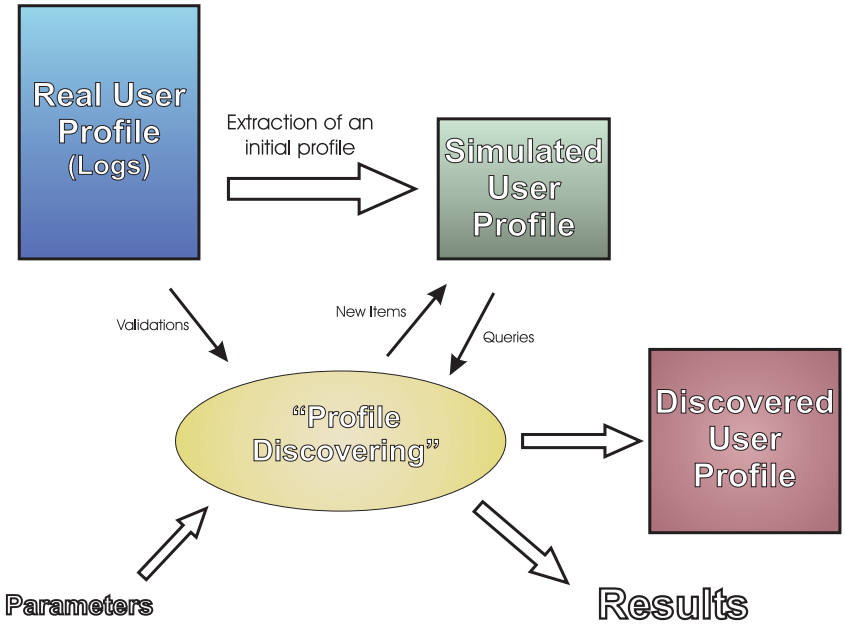


Figure 3.4: "Profile Discovering" Evaluation Procedure.

discovering procedure" (see Figure 3.4). This technique can be seen as an hybrid approach between real or laboratory evaluation, log analysis and user simulation.

First of all, it is necessary to obtain as many real user profiles as possible. These profiles must contain subjective assessments of the items (preferably explicit evaluations of the user, although the implicit information obtained from the user interaction with the system is also useful). It is desirable to obtain these user profiles through a real or laboratory evaluation although it implies a relatively long period of time. However, it is also possible and faster to get the user profiles through a questionnaire containing all the items which the users have to evaluate.

Once the real user profiles are available, the simulation process; that is the profile discovering procedure starts. It consists on the following steps:

1. Generation of an initial user profile (UP) from the real user profile (RUP, $UP \subset RUP$).

2. Emulation of the real recommendation process, where a new item (r) is recommended from the UP.

3. Validation of the recommendation:

- If $r \in RUP$, then r is considered as a discovered item and is added to UP ($UP = UP \cup \{r\}$).
- Otherwise, r is rejected

4. Repeat 2 and 3 until the end of the simulation.

As in the real evaluation, the simulation process starts with the generation of an initial user profile. It is desirable to initially know as much as possible from the user in order to provide satisfactory recommendations from the very beginning. Analysing the different initial profile generation techniques (described in section 2.3.2), namely: manual generation, empty approach, stereotyping and training set, we found different advantages and drawbacks. In manual generation, the user tailors his or her own profile, thus it is a really transparent method. But it bothers the user to have to do this and it is difficult for users to define their preferences explicitly. The empty approach needs potentially a long time to get to know the user's preferences; that is, the initial recommendations are low quality. But in this case, the user is not bothered. The usual approach is to interview the user with a quick manual questionnaire that won't annoy him or her too much, but people are reluctant to give personal data. Typically, the user does not fill in the questionnaire or provides false data. Stereotyping cannot be applied because our user profiles do not contain personal data and we cannot stereotype users. The training set approach depends totally on the profile learning technique (case retain in CBR), since the user just gives a list of items that he likes and/or dislikes, and the learning technique generates the profile. There is nothing to annoy the users and the users easily define their preferences. Therefore, in our approach, the training set seems to be the best technique for generating the initial profile (case base). Thus, the first step of the simulation consists in the extraction of an initial item set from the real user profile in order to generate the initial simulated user profile.

Then, the simulator emulates the recommendation of new items over time. In particular, it executes a process cycle by cycle, where a cycle is a day in the real world. During the simulated day, the recommendation algorithm recommends a group of items based on the information contained in the simulated user profile. All the functions, constraints and constants involved in the recommendation process are parameters of the simulator (for example, the time of the simulation, the recommendation algorithm or the learning parameters).

After each recommendation the simulator checks its success. In order to do that, the user's assessments are needed. Instead of inventing them like the latest simulators do, the profile discovering simulator looks up the real user profile containing the real evaluations of the user. Thus, if the item is contained in the real user profile, the simulator can check the user's opinion and classify the recommendation as a success or a failure accordingly. This discovered item is added to the user profile and a new item is recommended.

Once the simulation has been finished, the initial simulated profile will have evolved in a more complete profile that is called the discovered user profile. Note that the discovered profile can or cannot be equal to the real user profile, depending on how many items have been discovered. Moreover, the method provides results on how many recommendations have been made or how many successful/unsucessful items have been recommended. Thus, based on the discovered user profile and the final simulation results, different metrics can be evaluated.

The profile discovering procedure is a suitable instrument for evaluating recommender systems for several reasons:

- The recommendation process does not simulate the user evaluations, they are extracted from real user profiles.
- The simulation process considers the development of the user profile over time.
- Large-scale experiments are carried out quickly.
- Experiments are repeatable and perfectly controlled.

Developers can use the profile discovering evaluation procedure in order to test and compare their recommendation algorithms in order to find out which ones perform best. It is also a suitable instrument for tuning the parameters of a recommendation algorithm in order to obtain the best performance.

Cross-Validation through Profile Discovering

An extension of the profile discovering procedure has been implemented in order to perform cross-validations. Cross-validations are used to evaluate the accuracy of a recommender system. They consist of predicting the relevance of the examples

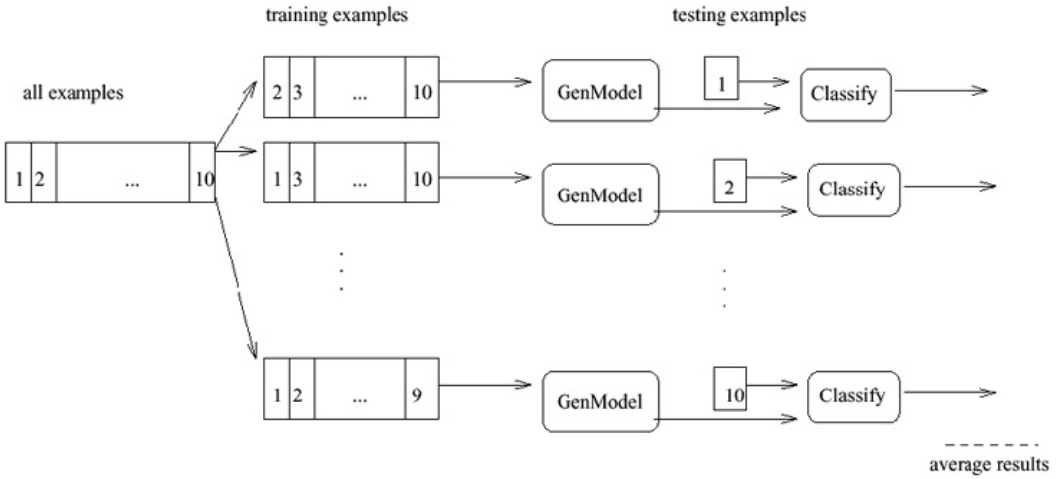


Figure 3.5: "10-fold cross-validation technique".

recorded in the logs and comparing them with the real evaluations. For example, the "10-fold cross-validation technique" [Mladenic 96], from a user profile with n items, generates n training sets of *n-1* items and, based on these, predicts the relevance of the reserved item (see Figure 3.5). Finally, the average of the precision of the predictions is the accuracy of the recommender system. The more precise the predictions are, the more accurate the system will be.

However, when the bulk of the recommender process remains in the adaptation that the recommendation algorithm performs on the profile over time, the "10-fold cross-validation technique" is useless. Before the cross-validation analysis, we need to know how the simulated user profile has evolved. We propose a cross-validation analysis through profile discovering. The idea is the same as the "10-fold cross-validation technique" in which we use the profile discovering procedure in order to simulate the user profile evolution (see Figure 3.6).

In particular, a single cross-validation performs the following steps:

1. Like the first step in profile discovering, a simulated initial profile is extracted from the log of the real user profile.

2. 10% of the items not contained in the simulated initial profile are randomly reserved as test items in order to perform the final cross-validation.

3. From the simulated initial profile, the profile discovering procedure is exe-

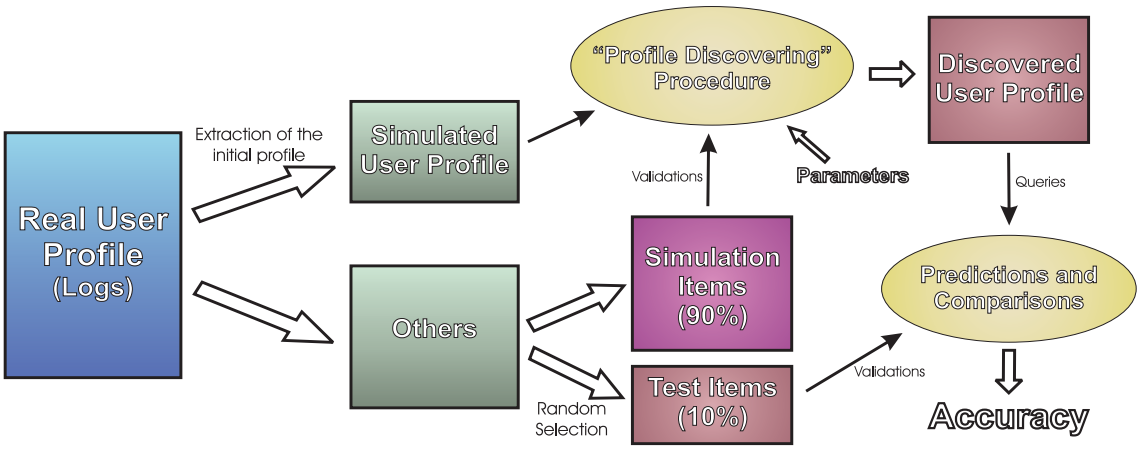


Figure 3.6: Cross-Validation through Profile Discovering.

cuted. The simulator has to discover the well evaluated items contained in the remaining 90%.

4. From the discovered user profile, a prediction of the relevance of the reserved test items is performed. Comparing the predictions with the real evaluations, the accuracy of the system is obtained.

In order to avoid anomalous data, the cross-validation is repeated many times, each time changing the test items (see Figure 3.7). From the real user profile, θ sets of test items are selected and, therefore, θ single cross-validations through profile discovering are performed. Two different θ values have been tested: $\theta = 10$ and $\theta = 20$. After several cross-validations we found that the results for both values were identical. Therefore, only the results with $\theta = 10$ are presented. The average of these simulations is the accuracy of the system.

3.6 Experimental Results

The proposed forgetting mechanism has been implemented in GenialChef[1], a restaurants recommender system developed within the IRES Project[2]. In particular, Ge-

[1]GenialChef was put forward at the E-TECH 2003 Prizes and was awarded the prize for the best university project.

[2]The IRES Project was presented to the AgentCities Agent Technology Competition and was awarded the special prize for the best system deployed in the AgentCities Network. Visit http://arlab.udg.es for more information about the project and the prize.

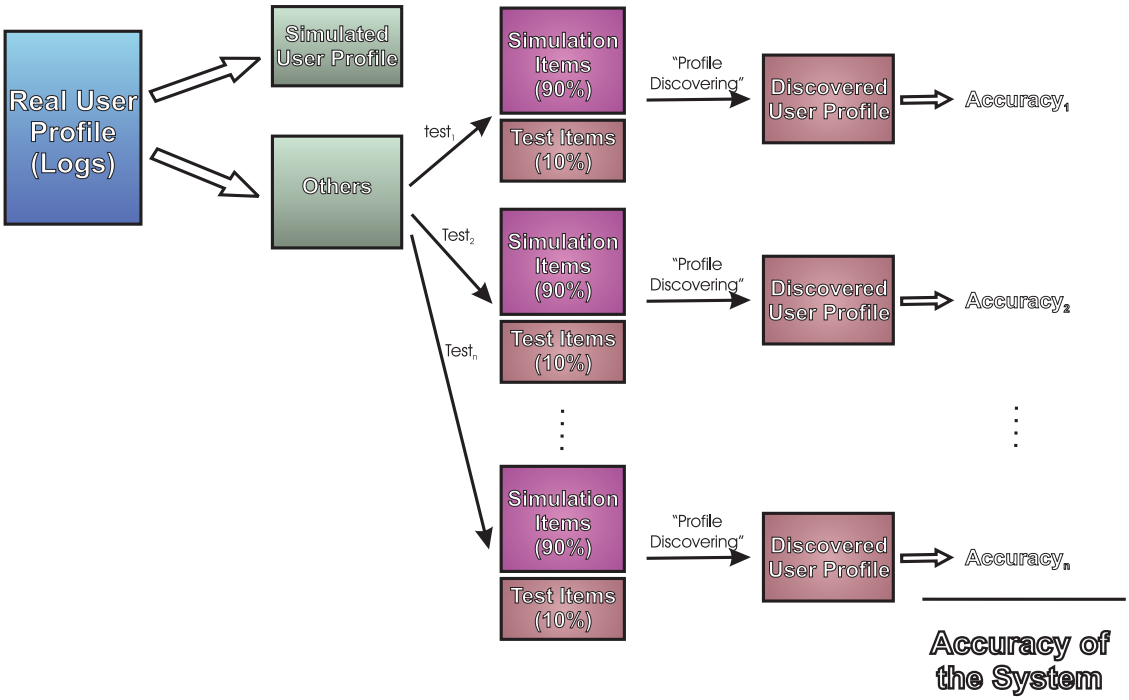
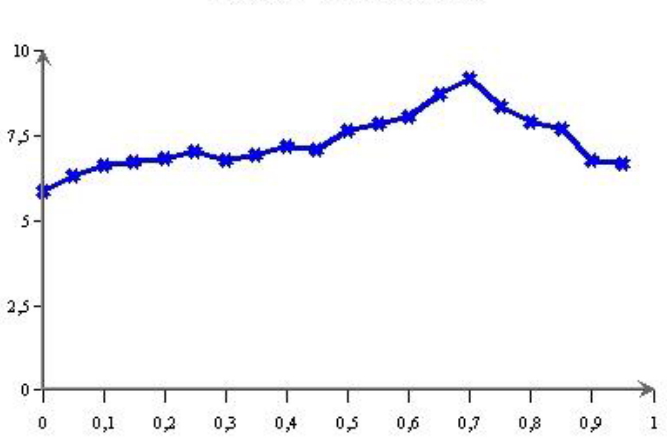


Figure 3.7: The Complete Cross-Validation Procedure.

nialChef provides a restaurant recommendation service to users from all over the world. So far, nearly 400 restaurants in and around Girona are recommended, thus, this is an interesting service for people living or visiting Girona.

The system was evaluated using the profile discovering procedure. We were able to repeat experiments with different parameters in order to show whether forgetting improves the quality of recommendations. With this aim in mind, all the evaluation measures explained in section 3.5.1 have been computed in relation to different degrees of forgetfulness.

The results of the simulations are analysed in this section with the help of several charts. The y-axis refers to the evaluation measures, namely precision, recall, f-measure, fallout, NCases, diversity and accuracy. All the measures, except diversity and NCases, are represented in percentages. The x-axis refers to the different levels of forgetfulness; that is, the drift threshold (see, for example, Figure 3.8). In particular, 20 different levels are analysed (every 0.05 from 0.0 to 0.95). The drift threshold, ξ, is the parameter used to control how much is forgotten, ranging from forgetting almost everything ($\xi = 0.95$) to forgetting nothing ($\xi = 0.0$). Note that if $\xi = 1$, the recommendations would be absolutely random, making nonsense of a



PARAMETERS [a]

CBR/E
6 *months*
*Training*0
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

[a]the parameters used in the simulation giving the chart on the left as a result.

Figure 3.8: Precision of the system.

recommender system. This simple representation allows us to find out the effect of the forgetting mechanism on each evaluation measure.

In order to avoid anomalous data, it is important to repeat the simulations with different input data. For this reason, 40 different real profiles were used in the simulations. In particular, 25 of these profiles were obtained after an evaluation period of 2 months where people from our department at the University of Girona used GenialChef. The remaining 15 profiles were obtained from a evaluation questionnaire filled in by people from outside of the University.

Below, we present the general results obtained with the profile discovering procedure. Then, these general results are validated, analysing the results obtained by changing the different parameters of the simulations.

3.6.1 General Results

In order to analyse the system performance from a global perspective, a set of simulation parameters were selected as representative. Specifically, the simulations were performed with the recommendation algorithm CBR/E; with the initial item set *training*0; with the forgetting parameters $\beta = 0.98$; $\lambda = 1.05$ and $\sigma = 0.95$; with the success and failure thresholds $\phi = 0.5$ and $\gamma = 0.499$ and finally, a 6 month simulation period. These parameters are described in detail in section 3.6.2.

Using these simulations, we analysed precision, recall, f-measure, fallout, NCases,

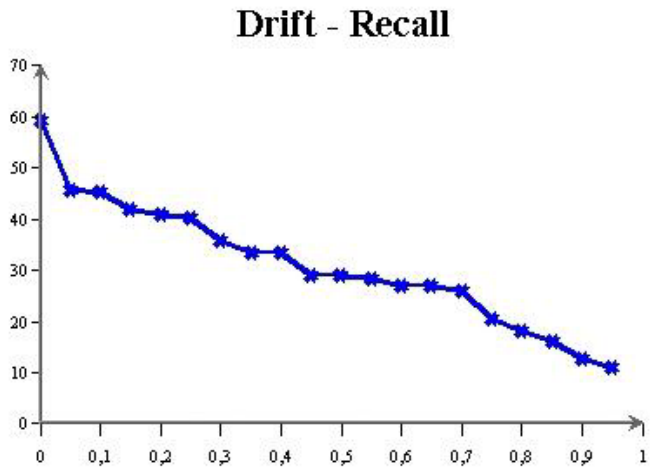


Figure 3.9: Recall of the system.

diversity and accuracy measures, the details of which are given below.

Precision

Figure 3.8 represents the precision of the system while taking into account different levels of forgetfulness. It is a bell-shaped graph, the maximum precision of which is found at 0.7 degree of forgetfulness. Note that the precision of the system is always higher with the forgetting mechanism ($\xi = 0.05 - 0.95$) than without ($\xi = 0$).

These results lead us to an interesting, although expected, result: the precision of the system is improved with the forgetting mechanism and the best precision is obtained when the recommender system forgets substantially. Furthermore, forgetting at any level gives a higher precision than not forgetting. However, forgetting too much is not good, since the precision is reduced.

Recall

Figure 3.9 represents the typical behaviour of recall in a recommender system when analysing different levels of forgetfulness. The highest level of recall is achieved when the system does not forget at all and, in the measure that the level of forgetfulness increases, recall descends. This behaviour has a logical explanation. On the one hand, with the highest forgetfulness, the system does not remember which items have been recommended and, therefore, it always recommends the same group of

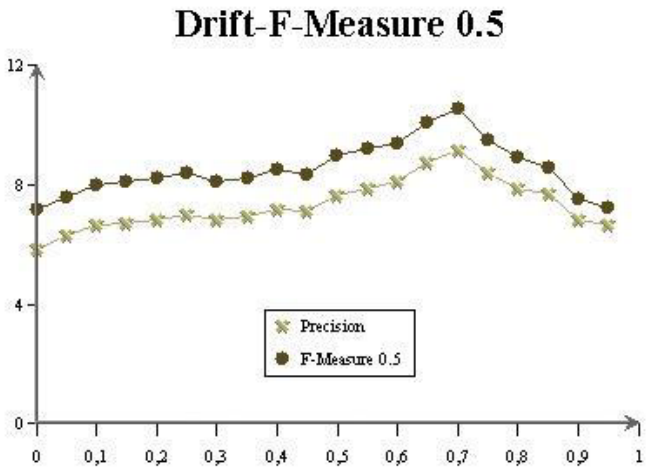


Figure 3.10: F-Measure of the system when b=0.5.

items. On the other hand, when the system does not forget at all, it always remember the past recommendations and their success/failure and, based on them, the system recommends new items.

Forgetting the non-relevant items is very productive in order to improve precision and in order to decrease the computing time. However, when the algorithm forgets too much, the same recommendations are repeated and, for this reason, we believe that forgetting excessively is unprofitable.

F-Measure

Figure 3.10 shows the f-measure when $b = 0.5$; that is, when precision is twice as important as recall. We can see how precision is slightly modified by recall, since when recall is very high, precision is rewarded and when recall is very low, precision is penalised.

Figure 3.11 shows the f-measure when $b = 1.0$; that is, when precision and recall are equally weighted. In this figure we can see a fairly strong modification of precision due to recall. When the system has a low level of forgetfulness, precision increases due to high recall, and when the level of forgetfulness is high, precision quickly decreases. However, the highest level of precision is still concentrated at around $\xi = 0.7$. This demonstrates the relevance of the forgetting mechanism.

It is also important to note that the lowest value of f-measure is obtained when

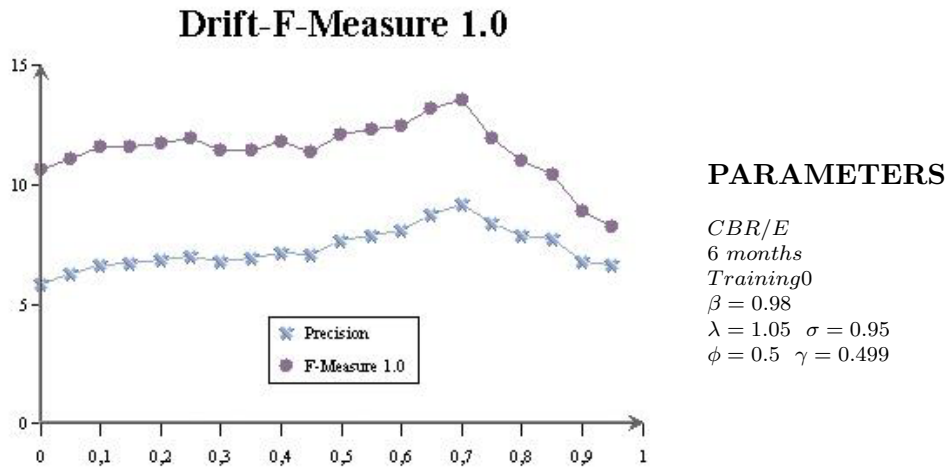


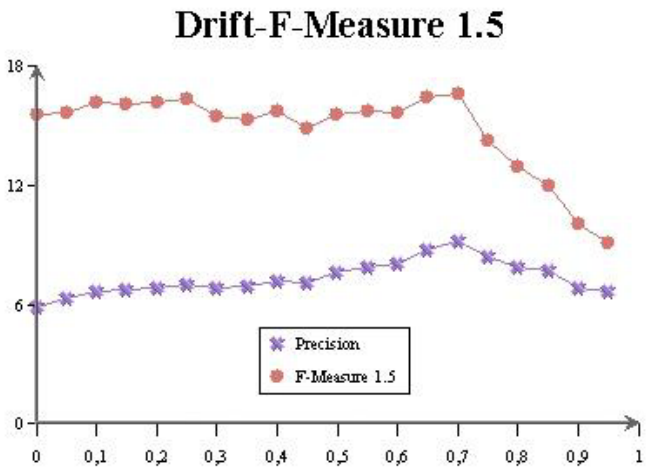
Figure 3.11: F-Measure of the system when b=1.0.

the system has a high level of forgetfulness, verifying that forgetting is productive, but forgetting too much is unfavourable. This phenomenon is present in all the f-measure 1.0 charts on all the experiments performed, since all the simulations have a very low recall when forgetting very much.

Figure 3.12 represents the f-measure when $b = 1.5$; that is, when recall is more important than precision. In other words, precision is strongly modified by recall. When the algorithm forgets moderately, the f-measure augments in such a way that its value remains stable. The chart still has the highest value when $\xi = 0.7$, the forgetfulness degree on which we obtain the best results; after this point, the performance of the system drastically decreases.

Fallout

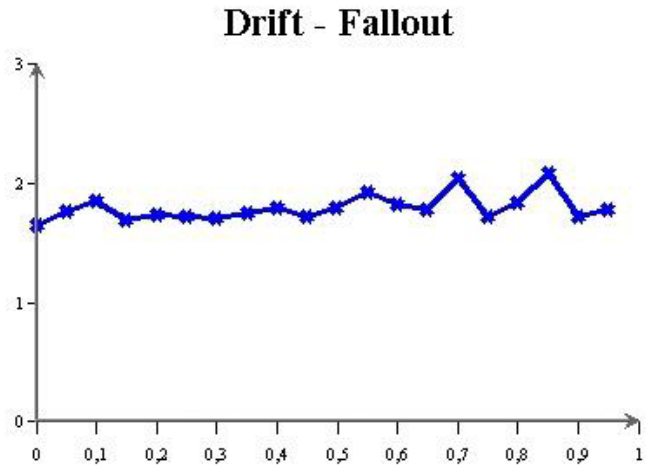
Figure 3.13 represents the fallout of a recommendation algorithm when the forgetting mechanism is applied. It appears to be quite stable, with fallout ranging from 1.6 to 2.1. Thus, a priori, we can conclude that the forgetting mechanism increases precision while preserving fallout. In any case, fallout is not as stable as desired. Hence, it is interesting to analyse the fallout of other simulations using different parameters in order to validate this result (see section 3.6.2).



PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.12: F-Measure of the system when b=1.5.



PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.13: Fallout of the system.

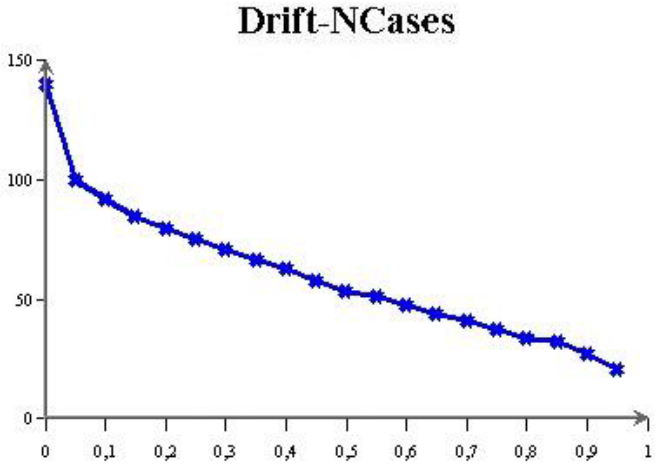


Figure 3.14: NCases of the system.

NCases

Figure 3.14 represents the average number of cases contained in the case base (NCases) throughout the simulations, with the parameters specified on the right of the figure. When the algorithm does not forget at all, there is a certain number of cases in the case base. Once the forgetting mechanism is applied ($\xi = 0.05$), a sizeable reduction in cases is observed between 25% and 30%. Then, proportionally to increasing forgetfulness, the NCases decrease. Finally, when the algorithm forgets almost everything, the NCases is reduced by between 80% and 90%. However, we discourage a very high level of forgetfulness. As seen in the previous charts, the best performance is obtained when forgetting substantially ($\xi = 0.7$). In this case, the number of cases if reduced by between 70% and 75%. This large reduction in cases in the case base is a relevant contribution to solving the utility problem of case-based reasoning systems [Leake 98].

The NCases results can be combined with the number of cases development over time, as shown in Figure 3.15. The number of cases are calculated from the simulations where each line represents a different level of forgetfulness. Initially, all the lines begin at the same number, which is the number of cases inserted by the initial profile generation technique. Then, depending on the level of forgetfulness the number of cases increases more or less. Obviously, the highest number of cases is obtained when the system does not forget at all ($\xi = 0.0$). After a certain period of time, depending on the level of forgetfulness, all the lines stabilise at a particular

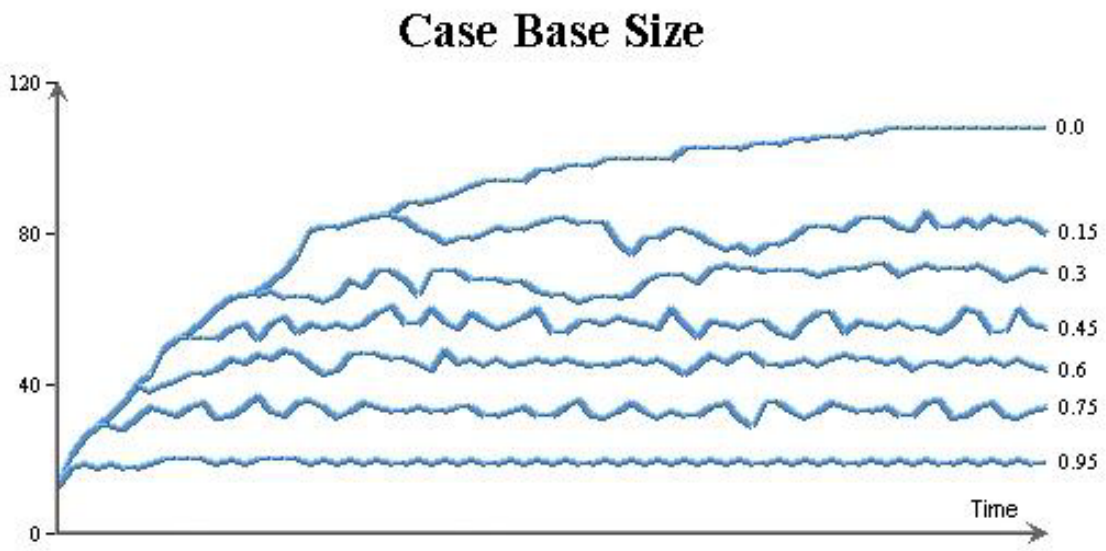


Figure 3.15: Evolution of the Number of Cases in the Case Base for different levels of forgetfulness.

number of cases.

Finally, notice that NCases is strongly correlated to the recall of the algorithm. Since all the recommendations are stored in the case base in order to know as much as possible about the user, if the recommendation algorithm has a high recall, the NCases is very high.

Diversity

In order to analyse the diversity of the user profile, we used the clustering method proposed in section 3.5.1. The first task that we have to perform is the selection of a suitable α. Taking into account that the final number of clusters represents the number of different classes of restaurants, what we have done is to apply the clustering method to the original database (where all the possible recommendations are contained) with different values of α. The results of the test are shown in the following table:

α	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5	0.55	0.6
Clusters	147	95	65	44	30	17	10	6	4	3	2	1

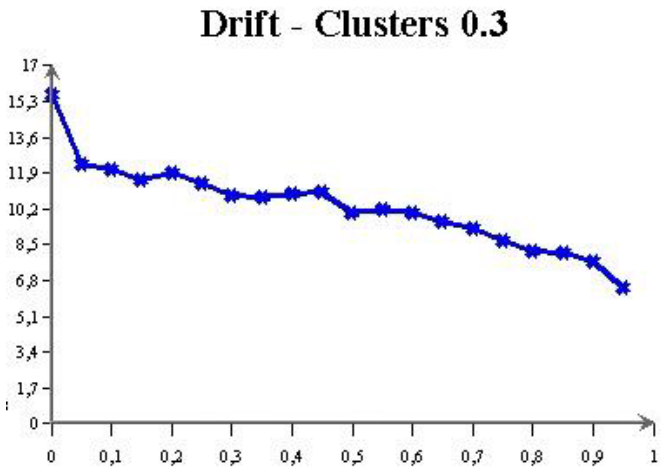


Figure 3.16: Diversity of the system.

If we assume that there are between 15 and 20 different classes of restaurants in the database[3], we select the $\alpha = 0.3$, since it has the number of clusters inside this interval. Thus, all the diversity charts are analysed with $\alpha = 0.3$.

Figure 3.16 presents the diversity of the recommendation algorithm with different levels of forgetfulness when $\alpha = 0.3$. There is a reduction in diversity as the algorithm forgets more. The diversity when the algorithm does not forget at all is almost the maximum, taking into account that the number of different classes is, at most, 17. This is a good result if what we are looking for is user profiles as diverse as possible. When we obtain a better precision (around $\xi = 0.7$), the number of clusters decreases to 10. Hence, 40% of the clusters have been lost. The resulting profile has less clusters, but they represent the user interests better. This is mainly due to the adaptation process that the drift attribute performs over the profile. In other words, the forgetting mechanism deletes the classes of items that do not contribute to the recommendation of interesting restaurants, thus obtaining a better precision.

In fact, the number of clusters depends on the user. There are users that have very select interests or users that do not have particular preferences and like almost everything. The first class of users tend to have a small number of clusters in their user profiles and the second ones tend to have a lot. However, these minor special classes of users are compensated by the majority.

[3]In Girona, a small city with less than 400 restaurants, this is a reasonable assumption.

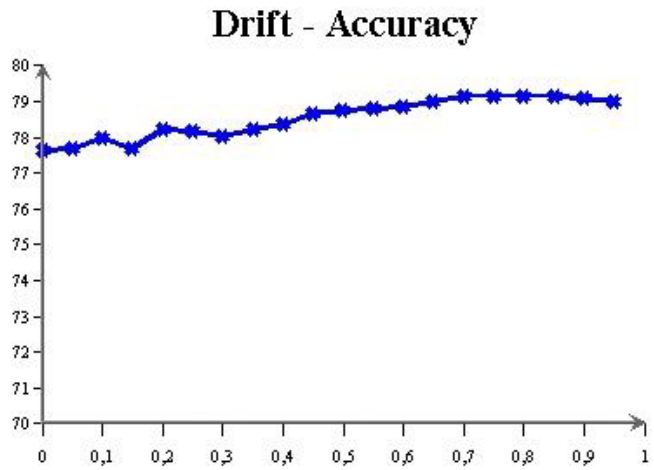


Figure 3.17: Accuracy of the system.

Accuracy

Finally, Figure 3.17 shows the accuracy of the system after performing several cross-validations with different levels of forgetfulness. Accuracy also produces a bell-shaped graph, although the increments and decrements are very small. The difference between the lowest accuracy (when the algorithm does not forget) and the highest accuracy (when the drift threshold is 0.75) is 1.5, an insignificant difference taking into account that our representation of the accuracy ranges from 0 to 100. Thus, we cannot say for certain that accuracy is improved by the forgetting mechanism; we can, however, say that there is no negative impact.

3.6.2 System Parameter Evaluation

In order to validate these general results, more than 150.000 simulations and 30.000 cross-validations with different user profiles, different forgetting parameters, different recommendation algorithms, different initial item sets and different simulation durations were performed. The parameters used in the simulations are summarised in Table 3.1 and the results of the simulations are shown in the following sections.

PARAMETERS	VALUES
User Profiles	$Profile_1$, $Profile_2$, ..., $Profile_{40}$
Drift Thresholds	$\xi = 0.0$, $\xi = 0.05$, $\xi = 0.1$, ..., $\xi = 0.95$
Recommendation Algorithms	CBR/E, CBR/I, $CBR/E(1)$, $CBR/I(1)$, $CBR/Random$
Drift Decreasing Factors	$\beta = 0.95$, $\beta = 0.97$, $\beta = 0.98$, $\beta = 0.99$
Success and Failure Thresholds	$\phi = 0.5$ $\gamma = 0.499$, $\phi = 0.6$ $\gamma = 0.4$
Initial Item Sets	$Training0$, $Training1$, $Training2$, $Training3$
Drift Rewarding and Penalising Factors	$\lambda = 1.1$ $\sigma = 0.9$, $\lambda = 1.05$ $\sigma = 0.95$
Simulation Durations	1 *month*, 6 *months*, 1 *year*, 2 *years*

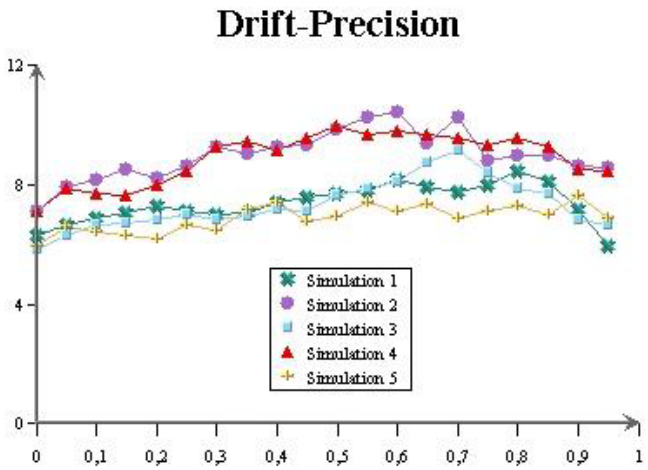
Table 3.1: Simulation Parameters.

Recommendation Algorithms

It is very important to validate the general results with several recommendation algorithms in order to demonstrate that the forgetting mechanism does not depend on them. Therefore, five different recommendation algorithms based on case-based reasoning (CBR) were implemented:

- CBR/E: a CBR algorithm that recommends several new items based on the most similar and the most different items to a set of the most liked and the most hated items correspondingly.
- CBR/I: a CBR algorithm that uses the implicit feedback (captured from the interaction between the user and the recommender system) to modify the user's evaluations. These new evaluations are used to decide the set of most liked and the most disliked items, the basis of the recommendation process.
- $CBR/E(1)$: a variation of the CBR/E algorithm which only gives 1 recommendation per cycle based on the most liked item.
- $CBR/I(1)$: a variation of the CBR/I algorithm which only gives 1 recommendation per cycle based on the most liked item.
- $CBR/Random$: selects an item from the user profile randomly and recommends the items that are most similar. That is, one of the user interests of the user is selected randomly ignoring their relevance and, based on it, similar items are recommended.

We expect these algorithms to behave in different ways. For example, the CBR/I algorithm is very similar to CBR/E, and it should recommend more items to the



PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

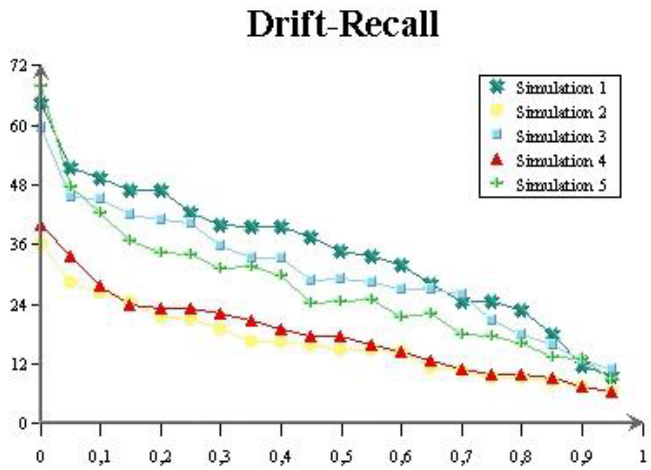
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.18: Precision, with different recommendation algorithms.

user since the implicit feedback contributes to finding out more diverse interests about the user and, therefore, the algorithm has more information based on which to recommend. The results of these algorithms give us, then, the possibility of evaluating whether the implicit feedback captured from the user will improve the performance of the system. $CBR/E(1)$ and $CBR/I(1)$ are algorithms from which we expect a smaller number of recommended items than from the first two, although they should be more precise, since only the item which is most similar to the most relevant user interest is recommended. Finally, the random algorithm, instead of recommending new items based on the most relevant interests of the user, recommends new items based on any of the user interests. Thus, we expect a lower precision, although not very low since the user interests are also taken into account.

Figures 3.18 to 3.26 show the results of the evaluation measures for each recommendation algorithm.

Precision. First, the behaviour of precision, Figure 3.18, shows that all the algorithms, except $CBR/Random$, have a bell-shaped graph, with the maximum precision at drift threshold ranges between 0.5 and 0.9. The precision of the $CBR/Random$ algorithm is stable, at between 6.5 and 7.5, through all the different levels of forgetfulness without distinction. This is a perfectly reasonable result, bearing in mind that there is no criteria for selecting relevant items from the user profile; all items



PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

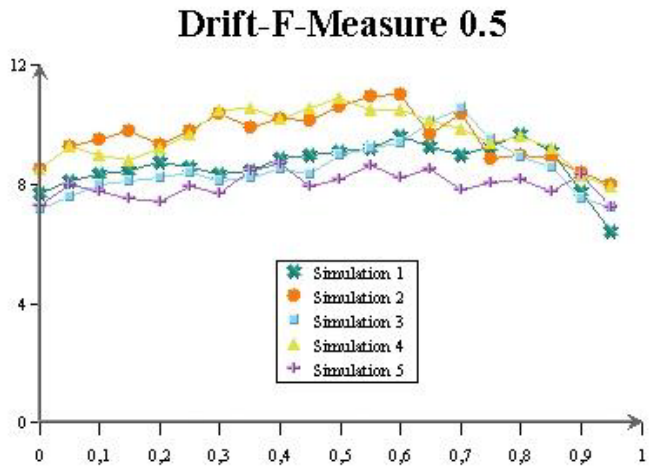
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.19: Recall, with different recommendation algorithms.

selected randomly. On the contrary, the other algorithms use the drift attribute in order to find out which are the most relevant. Hence, the $CBR/Random$ algorithm is the one with the lowest precision. As we expected, the recommendation algorithms with the highest precision are $CBR/E(1)$ and $CBR/I(1)$.

Recall. Second, Figure 3.19 shows that the recommendation algorithms have different recall values, although they follow the same pattern. The system achieves high recall value when the algorithm does not forget anything and a gradual decrease as the level of forgetfulness increases. As we expected, the recommendation algorithm with the highest recall is CBR/I and the ones with the lowest recall are $CBR/E(1)$ and $CBR/I(1)$.

F-Measure. Third, Figure 3.20, Figure 3.21 and Figure 3.22 analyse the f-measure of the recommendation algorithms. From the point of view of precision and recall together, the algorithms all perform similarly. Note that the algorithms that recommend a new item based on only the most liked item ($CBR/E(1)$ and $CBR/E(1)$) have a higher precision than the others. However, we can also see that they are the ones with the lowest recall. This behaviour corroborates the fact that when recall increases, precision decreases and, therefore, the f-measure compensates them, resulting in similar charts.

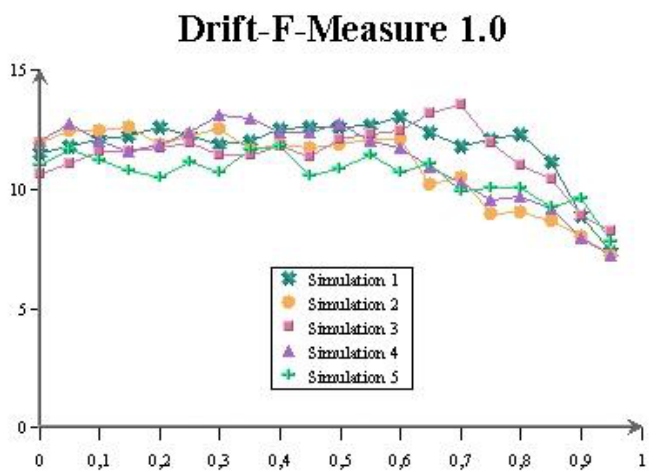


PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.20: F-Measure, with different recommendation algorithms when b=0.5.

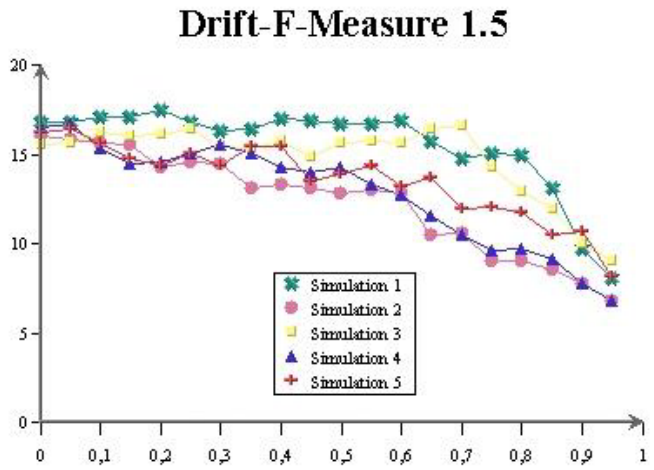


PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.21: F-Measure, with different recommendation algorithms when b=1.0.

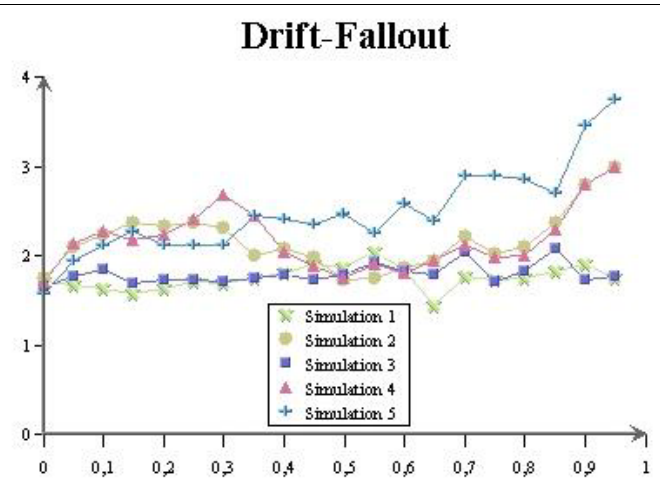


PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.22: F-Measure, with different recommendation algorithms when b=1.5.

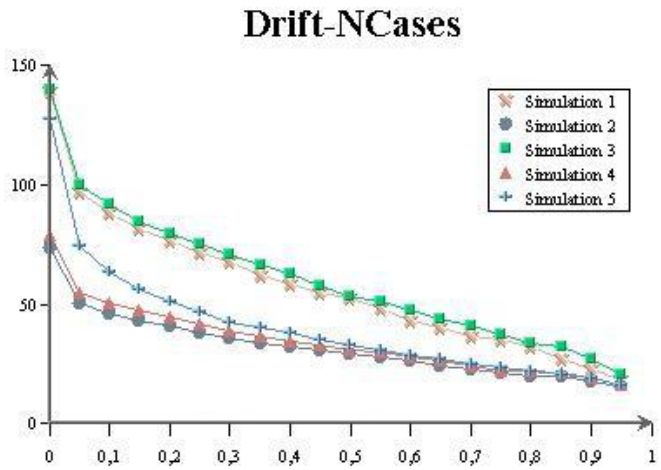


PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.23: Fallout, with different recommendation algorithms.



PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

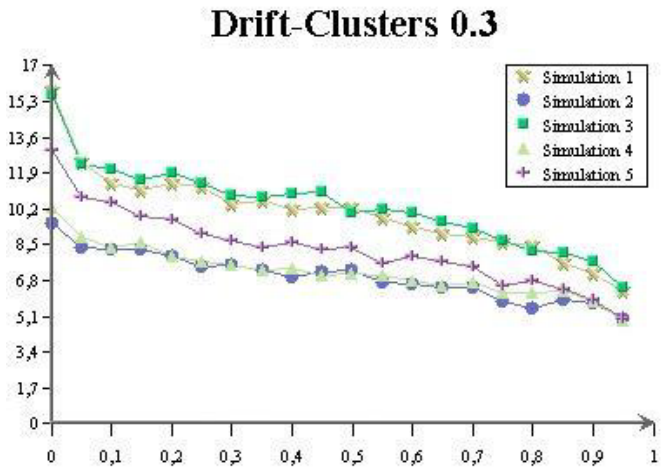
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.24: NCases, with different recommendation algorithms.

Fallout. Fourth, Figure 3.23 presents the fallout of the recommendation algorithms obtained when different levels of forgetfulness are applied. When no cases are forgotten, all the algorithms have the same fallout. However, the forgetting mechanism affects each of the algorithms in a different way. CBR/E and CBR/I behave as expected, since the fallout remains stable whatever the level of forgetfulness. $CBR/E(1)$ and $CBR/I(1)$ also have stable fallout but, when the algorithm forgets too much, fallout increases sharply. The worst fallout occurs when the system recommends with the $CBR/Random$ algorithm. In this case, fallout remains higher than for the others and, when the algorithm forgets a lot, fallout is extremely high. This is what we expect from a random algorithm.

NCases. Fifth, Figure 3.24 illustrates the average number of cases contained in the case base during the simulations with the different recommendation algorithms. All the algorithms, except $CBR/Random$, lose between 70% and 75% of the cases when $\xi = 0.7$. The $CBR/Random$ algorithm loses 81% of the cases, because cases are strongly penalised due to the lack of a selection criteria. Our results show that the forgetting mechanism drastically reduces NCases while increasing precision. And, as we can see in Figure 3.24, this behaviour is totally independent of the recommendation algorithm.

The algorithms that have the lowest NCases ($CBR/E(1)$ and $CBR/E(1)$) are



PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

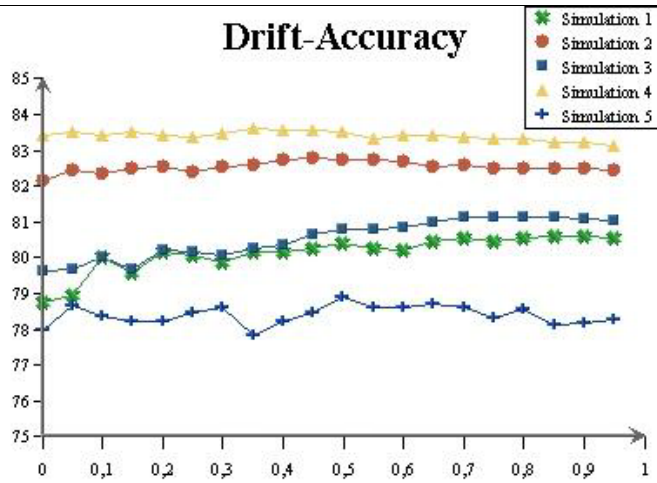
Figure 3.25: Diversity, with different recommendation algorithms.

also the ones with the lowest recall. Furthermore, CBR/E and CBR/I are the algorithms with the highest NCases and the highest recall. This shows that the NCases is strongly correlated to the recall of the algorithm.

Diversity. Sixth, the lost of diversity is considerable, whatever the recommendation algorithm (see Figure 3.25). The observed losses range from 32% with the $CBR/I(1)$ to 44% with the CBR/I. The algorithms that have the lowest loss of diversity are $CBR/I(1)$ and $CBR/E(1)$, precisely the ones that have the lowest recall and NCases. In contrast, the algorithms with the highest recall and NCases also have the highest loss of diversity. Hence, the correlation between loss of diversity and recall and NCases is obvious.

Accuracy. Finally, the results for accuracy (see Figure 3.26) with different recommendation algorithms are very similar to the ones obtained for precision. $CBR/I(1)$ and $CBR/E(1)$ are the algorithms with the best accuracy while $CBR/Random$ is the worst. However, the most important result is that the accuracy of the system remains stable whatever the recommendation algorithm.

Summing up all the experiments with regard to the different algorithms, we can say that the forgetting mechanism improves the precision of the system provided



PARAMETERS

Simulation1 - CBR/I
Simulation2 - $CBR/I(1)$
Simulation3 - CBR/E
Simulation4 - $CBR/E(1)$
Simulation5 - $CBR/Random$

6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.26: Accuracy, with different recommendation algorithms.

that the recommendation algorithm recommend new items which take into account the relevance of the user's interest contained in the user profile. The algorithms that only recommend the best item are the ones with the highest precision and accuracy, although they have the lowest recall. So, depending on the aim of the recommender system, a different recommendation algorithm can be selected. Furthermore, all the experiments show that CBR/E and CBR/I perform in a similar way. Therefore, we can also conclude that the incorporation of implicit feedback in the recommendation process does not produce significative improvements to the performance of the system.

Drift Decreasing Factors

The drift decreasing factor (β) is used to control the speed of forgetting. The higher it is, the lower is the speed. The speed of forgetting is strongly related to the forgetfulness threshold. That is, if we forget too fast and our forgetfulness threshold is high, the information on the user is reduced drastically and, with less information, the more difficult it is to provide good results. In order to test this hypothesis, we tested four different β values: $\beta = 0.95$, $\beta = 0.97$, $\beta = 0.98$ and $\beta = 0.99$. The impact of the β parameter on the different evaluation measures are shown in Figures 3.27 to 3.34.

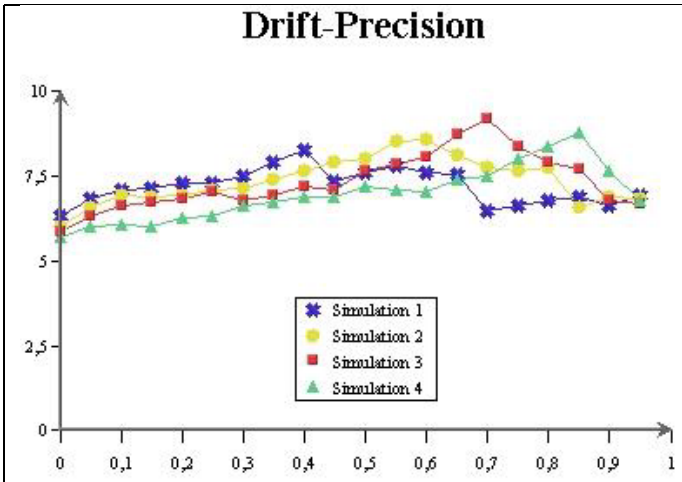
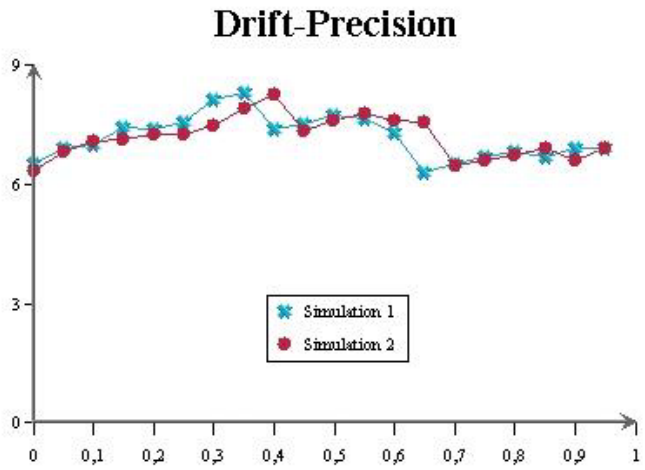


Figure 3.27: Precision of the system with different drift decreasing factors.

Precision. First, Figure 3.27 shows the precision of the system with the different drift decreasing factors. All the graphs are bell-shaped graph and, depending on the speed, the highest precision is found in the range of $\xi = 0.5 - 0.9$. However, when $\beta = 0.95$; that is, when there is a high speed, the graph loses the bell shape and becomes unstable. Moreover, the maximum precision is lower than it is for the other charts and not forgetting ($\xi = 0.0$) produces greater precision than forgetting substantially. In order to validate this strange behaviour, we designed and performed another set of experiments with high speed. Figure 3.28 shows the precision of the CBR/E algorithm when $\beta = 0.94$ and the precision graph has also a similarly strange shape. Despite these results, we can say that the speed of forgetting shifts the forgetfulness threshold. The slower the speed, the higher the forgetfulness threshold where the maximum precision is found.

Recall. Second, the speed of the forgetting mechanism also affects the recall of the recommendation algorithm (see Figure 3.29). We obtained the lowest recall when $\beta = 0.95$; that is, when the algorithm most rapidly forgets. This is mainly due to the fact that the algorithm forgets any recent recommendations and tends to recommend the group of items most similar to the user preferences, which are always the same. As the speed decreases, the algorithm remembers recent recommendations and tries to recommend different items and, therefore, recall increases. Thus, the highest recall is obtained when the algorithm has the lowest speed ($\beta = 0.99$).

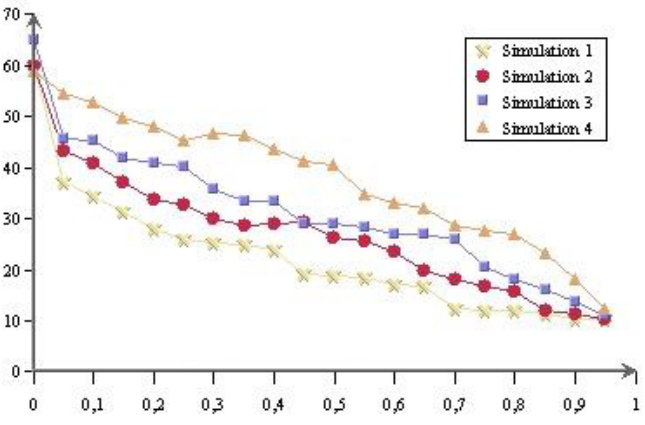


PARAMETERS

CBR/E
6 *months*
*Training*0
Simulation1 - $\beta = 0.94$
Simulation2 - $\beta = 0.95$

$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.28: Precision of the system with two low drift decreasing factors.

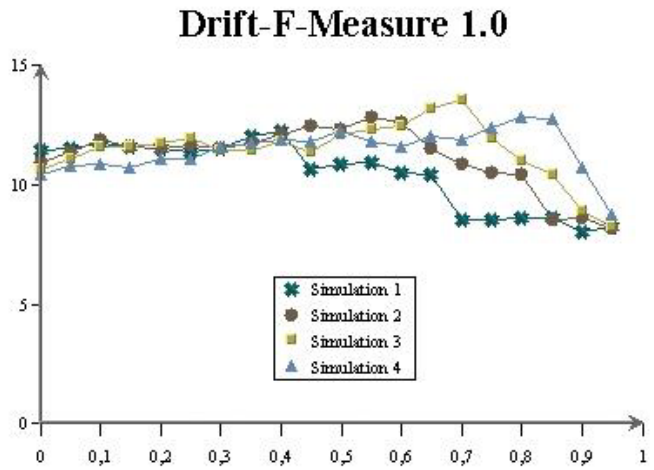


PARAMETERS

CBR/E
6 *months*
*Training*0
Simulation1 - $\beta = 0.95$
Simulation2 - $\beta = 0.97$
Simulation3 - $\beta = 0.98$
Simulation4 - $\beta = 0.99$

$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.29: Recall of the system with different drift decreasing factors.



PARAMETERS

CBR/E
6 *months*
$Training0$
Simulation1 - $\beta = 0.95$
Simulation2 - $\beta = 0.97$
Simulation3 - $\beta = 0.98$
Simulation4 - $\beta = 0.99$

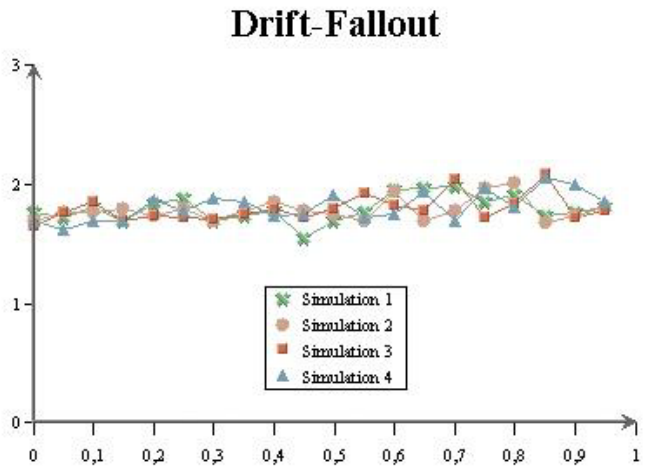
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.30: F-measure of the system with different drift decreasing factors when b=1.0.

F-Measure. Third, when analysing the f-measure 1.0 of the different drift decreasing factors (see Figure 3.30), we can see that the graphs are quite similar to the precision graphs, except when $\beta = 0.95$. Although it is the one with least recall, it does not have good precision, and this is too relevant for the f-measure.

Fallout. Fourth, Figure 3.31 illustrates the fallout of the recommendation algorithm. All the charts are similar. They are all stable and within the same small range of values. Therefore, we can say that the speed of the forgetting mechanism does not influence the fallout of the system, which is very stable at the different levels of forgetfulness.

NCases. Fifth, Figure 3.32 presents the NCases of the case base when different drift decreasing factors are applied. The speed of the forgetting mechanism also affects the number of cases contained in the case base; that is, the information about the user. On the one hand, when the algorithm has a high speed (for example $\beta = 0.95$), the number of cases is more quickly reduced and, therefore, the NCases value is lower. On the other hand, when the algorithm needs more time to forget, there are indeed more cases in the case base and the algorithm has a higher NCases value. These results, together with the precision results, corroborate our hypothesis that a higher speed reduces the information about the user.

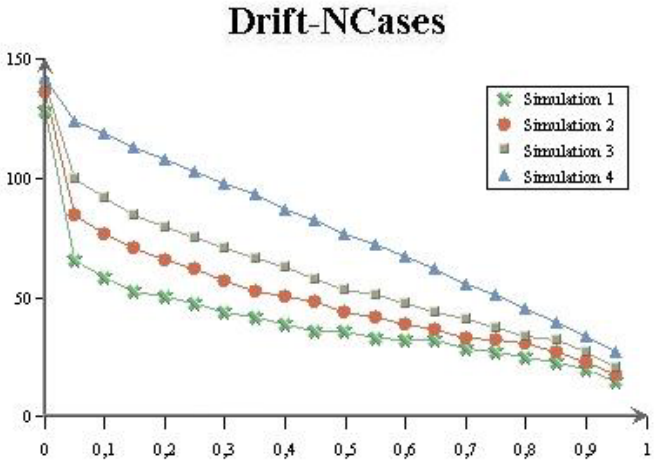


PARAMETERS

CBR/E
6 *months*
*Training*0
Simulation1 - $\beta = 0.95$
Simulation2 - $\beta = 0.97$
Simulation3 - $\beta = 0.98$
Simulation4 - $\beta = 0.99$

$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.31: Fallout of the system with different drift decreasing factors.



PARAMETERS

CBR/E
6 *months*
*Training*0
Simulation1 - $\beta = 0.95$
Simulation2 - $\beta = 0.97$
Simulation3 - $\beta = 0.98$
Simulation4 - $\beta = 0.99$

$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.32: NCases of the system with different drift decreasing factors.

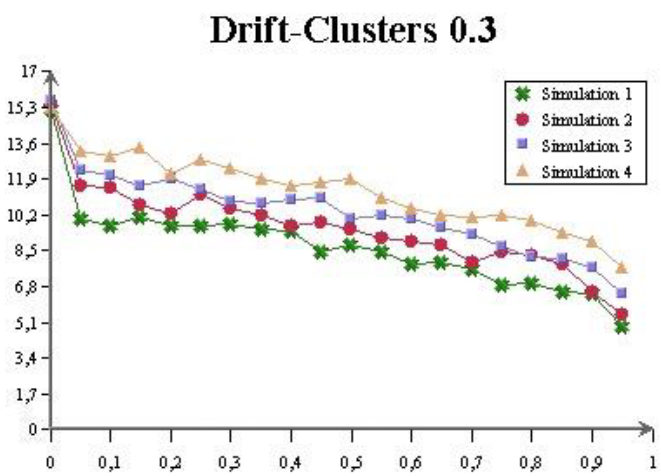


Figure 3.33: Diversity of the system with different drift decreasing factors.

Diversity. Sixth, the speed of the forgetting mechanism affects the diversity of the final user profile in the same way that it affects the average of cases in the case base (see Figure 3.33). Forgetting very fast means that cases need more time to be forgotten and, therefore, the diversity also remains for longer. This phenomenon can be seen clearly when we compare the 33% loss when the system forgets slowly with the 50% when the system forgets very quickly.

Accuracy. Finally, no significative differences can be observed from the point of view of the accuracy measure when different drift decreasing factors are evaluated (see Figure 3.34). This chart only verifies that the accuracy of the system remains stable regardless of the speed of the forgetting mechanism.

The analysis of the results of the different drift decreasing factors leads us to believe that forgetting is important, provided that the degree of speed is suitable. Forgetting with a high speed produces a lower, unstable precision. When different slow speeds are analysed, their behaviour is quite similar. As β is increased, the main differences are that the maximum precision is located closer to $\xi = 0.95$; recall and NCases increase and less diversity is lost.

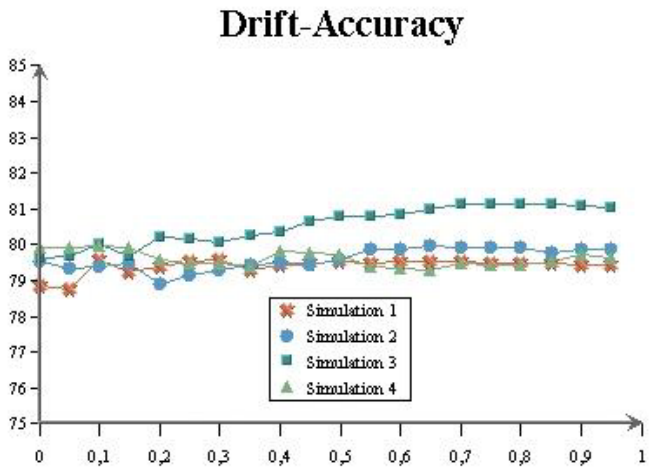


Figure 3.34: Accuracy of the system with different drift decreasing factors.

Success and Failure Thresholds

At the revision phase of the CBR approach to recommender system, the user evaluates the item proposed by the system (see Section 3.3). The evaluation of an item by a user is a value ranging from 0 to 1, where 1 represents total affinity to the item and 0 represents a total dislike to it. But, what do we consider as a successful recommendation and what is a failed recommendation? In order to avoid mere assumptions, we have provided the simulator with two parameters that control successes and failures:

- the success threshold (ϕ): when the evaluation of the user is over this threshold, the recommendation is considered a success.
- And the failure threshold (γ): when the evaluation of the user is under this threshold, the recommendation is considered a failure.

The success and failure thresholds depend on subjective assessments related to the functionality of the system. Our intention is to demonstrate that the forgetting mechanism does not depend on these assessments. To do so, we analysed two different success thresholds ($\phi = 0.5$ and $\phi = 0.6$) and two different failure thresholds ($\gamma = 0.499$ and $\gamma = 0.4$). Figures 3.35 to 3.40 show the results on the different evaluation measures.

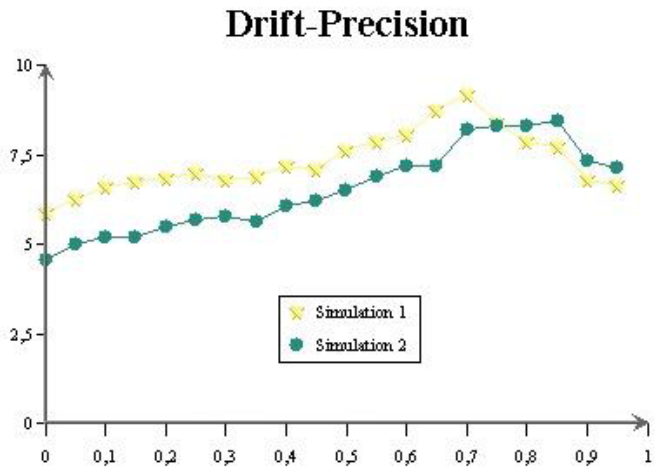


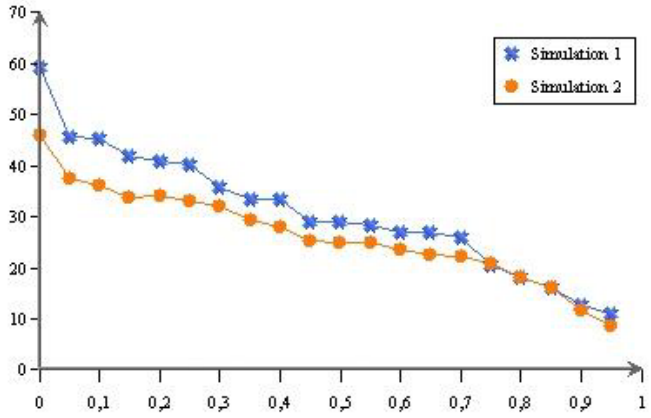
Figure 3.35: Precision of the system with different success/failure thresholds.

Precision. Figure 3.35 shows the precision of the system with the different success and failure thresholds. Both graphs are very similar. The main difference is quite logical: precision is lower when there is a range of uncertainty (between $\phi = 0.6$ and $\gamma = 0.4$), since several evaluated items are not considered as successful.

Recall. Likewise, when we analyse the recall of the system (see Figure 3.36), we can see that when $\phi = 0.6$ and $\gamma = 0.4$, recall is lower. This is mainly due to the fact that, when there is a range of uncertainty, a certain group of items cannot be classified and, therefore, the recommendation algorithm has less relevant information to use for recommending items, so the number of recommendations is lower.

Fallout. In the same way, fallout is also reduced when the range of failures is reduced. Figure 3.37 shows the fallout of the system with different failure thresholds. As expected, when $\gamma = 0.499$, the fallout is higher than when $\gamma = 0.4$. γ determines when a recommendation is considered a failure; so the higher the γ, the wider the range in which a recommendation is considered a failure and, therefore, the higher the fallout.

NCases. Then, since recall is lower when there is a range of uncertainty, NCases is also lower (see Figure 3.38). However, the difference between the two NCases

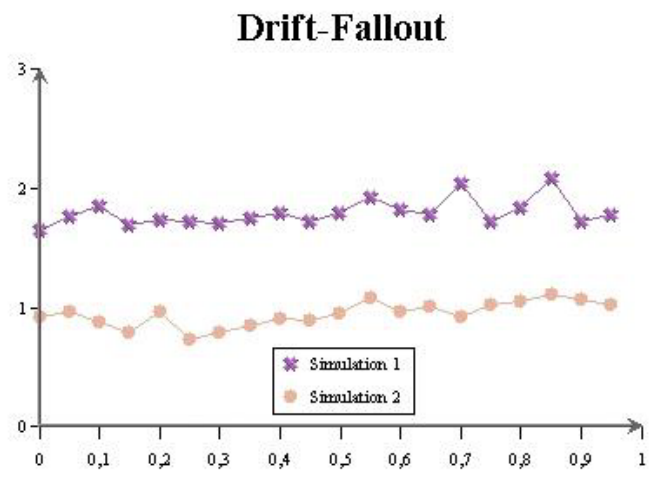


PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$

Simulation1 - $\phi = 0.5 \quad \gamma = 0.499$
Simulation2 - $\phi = 0.6 \quad \gamma = 0.4$

Figure 3.36: Recall of the system with different success/failure thresholds.

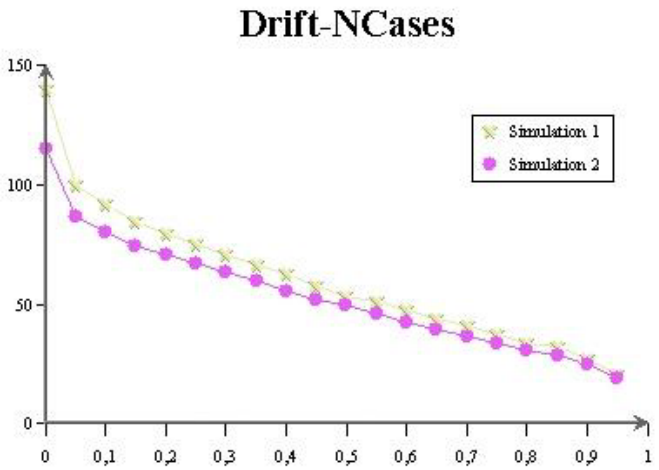


PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$

Simulation1 - $\phi = 0.5 \quad \gamma = 0.499$
Simulation2 - $\phi = 0.6 \quad \gamma = 0.4$

Figure 3.37: Fallout of the system with different success/failure thresholds.



PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$

Simulation1 - $\phi = 0.5 \quad \gamma = 0.499$
Simulation2 - $\phi = 0.6 \quad \gamma = 0.4$

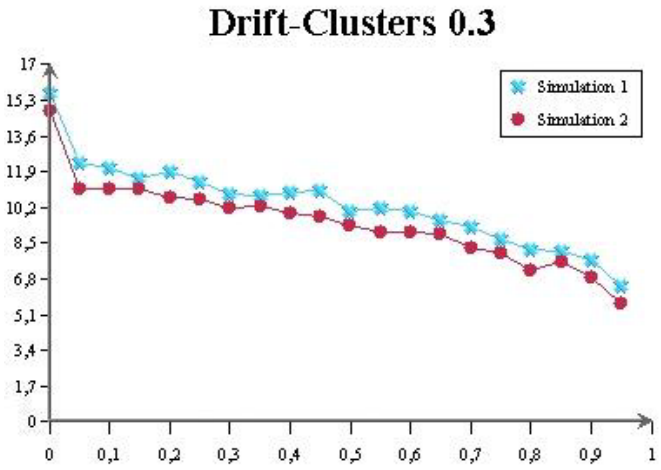
Figure 3.38: NCases of the system with different success/failure thresholds.

graphs is irrelevant due to the fact that the range of uncertainty between $\phi = 0.6$ and $\gamma = 0.4$ is small. If we increase this range, the difference should be more significant.

Diversity. There is no significative difference in the loss of diversity when different success and failure thresholds are analysed. We can see in Figure 3.39 that, when there is a range of uncertainty, there are less cases in the case base and, therefore, diversity is lower. However, the same percentage of diversity is lost.

Accuracy. Finally, Figure 3.40 illustrates the accuracy of the recommendation algorithm CBR/E when the different success and failure thresholds are simulated. Once again, the accuracy is lower when the range of success and failure is narrower. We can even observe that both simulations produce an almost identically shaped graph.

Summing up, the only effect obtained when we reduce the success and failure ranges is lower precision, lower recall, lower fallout, lower NCases and lower accuracy. But the system maintains the same expected behaviour, independently of the subjective assessments provided by the system.

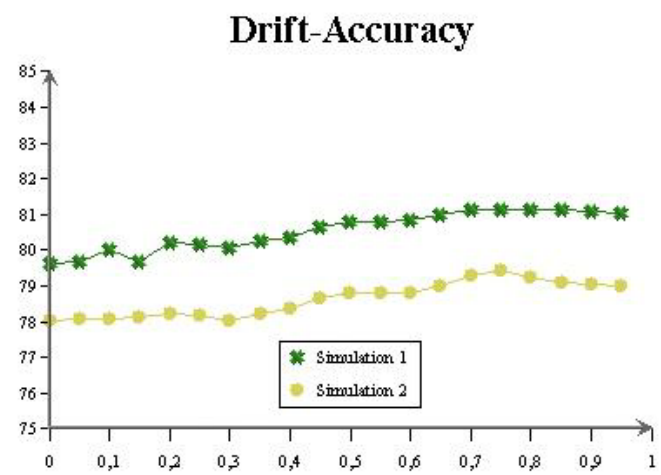


PARAMETERS

CBR/E
6 $months$
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$

Simulation1 - $\phi = 0.5 \quad \gamma = 0.499$
Simulation2 - $\phi = 0.6 \quad \gamma = 0.4$

Figure 3.39: Diversity of the system with different success/failure thresholds.



PARAMETERS

CBR/E
6 $months$
$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$

Simulation1 - $\phi = 0.5 \quad \gamma = 0.499$
Simulation2 - $\phi = 0.6 \quad \gamma = 0.4$

Figure 3.40: Accuracy of the system with different success/failure thresholds.

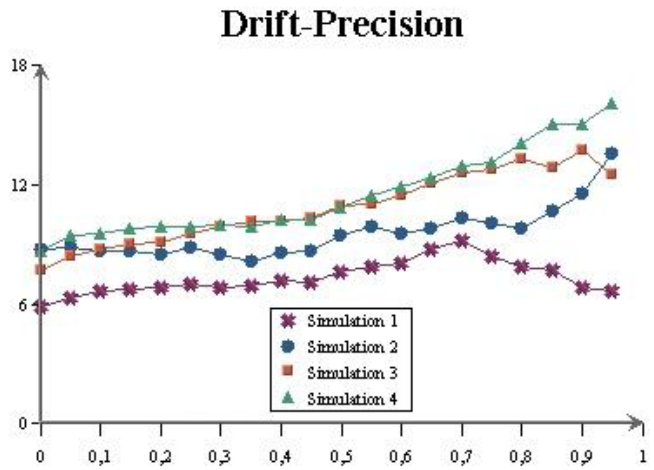


Figure 3.41: Precision of the system with different initial item sets.

Initial Item Sets

The set of items constituting the initial simulated user profile is a parameter of the profile discovering simulator. The aim of this parameter is to analyse the relationship between the forgetting mechanism and different initial profiles. Most learning algorithms are sensitive to the initial item set and CBR is not an exception. In our case, we are interested in the impact of the initial set of cases on the forgetting mechanism. Since we are trying to adapt the recommender system to the user interests, we believe that the impact does not depend on the number of cases but on their quality.

In order to test our hypothesis, we defined four different item sets:

- *training*0: with 13 representative items,
- *training*1: with 2 items, the most and the least liked.
- *training*2: with 5 representative items,
- and *training*3: with 6 items, the three most and the three least liked.

Figures 3.41 to 3.48 display the experimental results achieved with the different initial item sets.

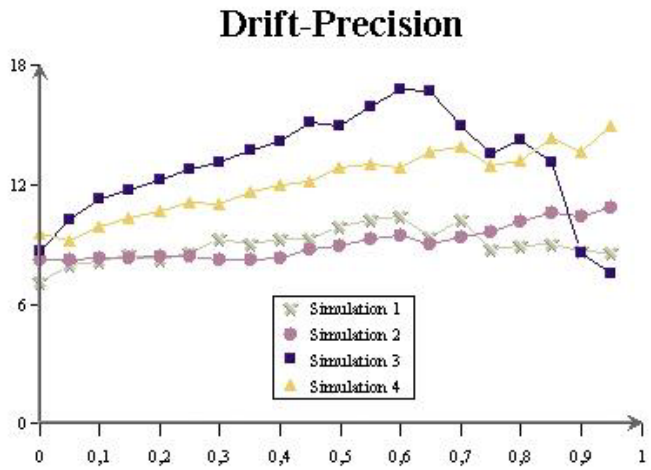


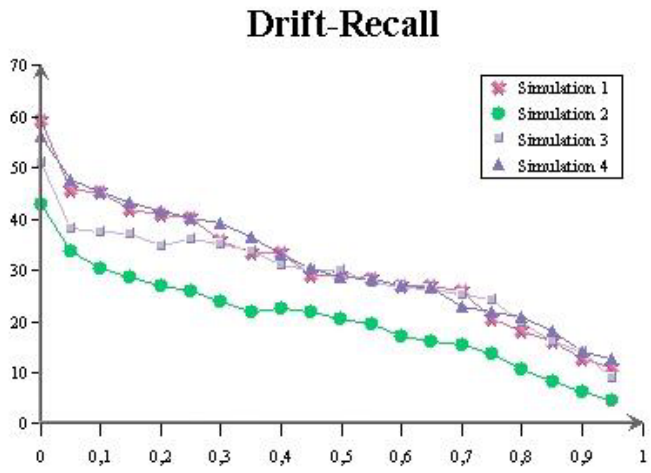
Figure 3.42: Precision of the system with different initial item sets and the $CBR/I(1)$ algorithm.

Precision. First, Figures 3.41 and 3.42 show the precision of the system according to the different initial item sets. We can see that the results are very different, giving us an idea of the importance of the initial item set in the performance of the system.

Depending on the synergy between the recommendation algorithm and the initial item set, the precision of the system changes considerably. For example, when only a few items constitute the initial case base (for example, *training*1, *training*2 and *training*3), we can see two different patterns of behaviour:

- On the one hand, when the items are very well selected according to the specific recommendation algorithm, the initial recommendations are very successful and consequently, the initial precision is very high. When the algorithm forgets rapidly, the same recommendations are repeated and the precision remains very high. Then, when the level of forgetfulness decreases, the algorithm gradually loses precision. This behaviour can be observed for $Training1$, $Training2$ and $Training3$ in Figure 3.41, and for $Training1$ and $Training3$ in Figure 3.42. We also see that the recall measure is very low and, therefore, when we combine precision and recall, the graphs regain the typical bell shape observed in previous simulations.

- On the other hand, when the initial items are unsuitable for the recommendation algorithm, the initial recommendations are unsuccessful and precision



PARAMETERS

CBR/E
6 *months*

Simulation1 - *Training*0
Simulation2 - *Training*1
Simulation3 - *Training*2
Simulation4 - *Training*3

$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.43: Recall of the system with different initial item sets.

is very low. Therefore, when the system forgets a lot, the same unsuccessful recommendations are repeated and precision remains very low. However, as forgetfulness decreases, the system remembers its bad performance, learns from its errors and tries to find new recommendations. Therefore, precision improves sharply. Finally, when there is a very low level of forgetfulness the algorithm loses precision. Hence the precision of the system produces an exaggerated bell-shaped graph. This is the case for *Training*2 in Figure 3.42.

A very different case is when a relatively high number of items constitute the initial case base (for example, the *training*0 set). In this case, there are more items in the case base and the algorithm has more information on which to base its recommendations. Having recommended more heterogeneous items, it is more difficult to obtain success for each of them. For this reason, with a high degree of forgetfulness, precision is lower than the other sets. However, when the algorithm forgets less, it knows better which are the relevant items and precision increases significantly. Finally, the precision tends to diminish again when the level of forgetfulness decreases. This behaviour is observed for *Training*0 in Figures 3.41 and 3.42.

Recall. Second, the system exhibits a similar recall performance when applying the different initial item sets except for *Training*1 (see Figure 3.43). This is because it is the smallest initial item set with only two cases constituting. Since there is only

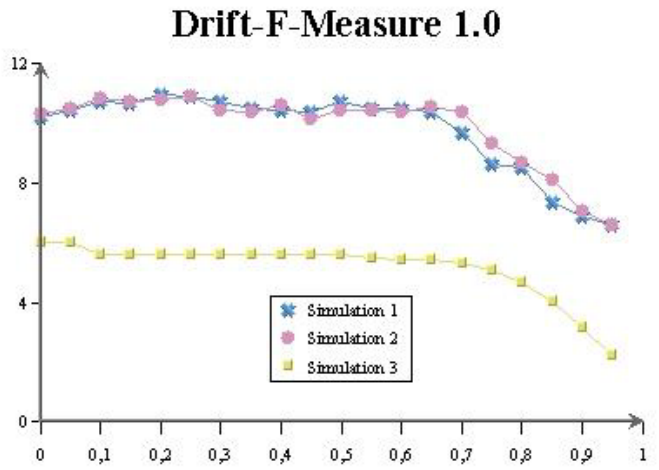


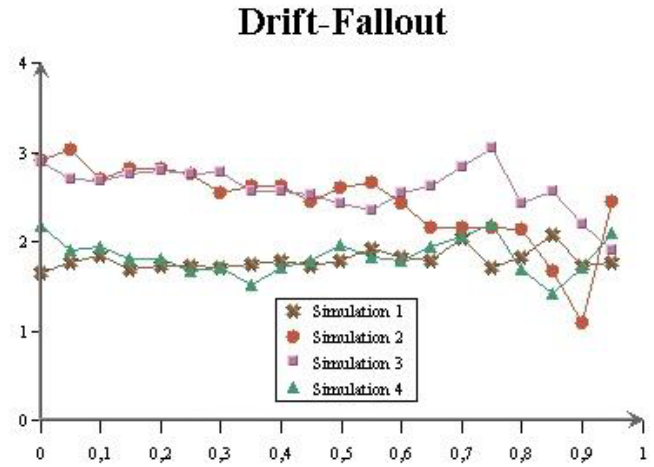
Figure 3.44: F-Measure of the system with different initial item sets when b=1.0.

a very little information on which to base the recommendations, the algorithm is not able to find many new interesting items and, therefore, the recall of the algorithm is very low. On the contrary, when a significant number of cases constitute the initial profile, the algorithm is able to recommend much more items.

F-Measure. Third, the f-measure of the different initial item sets deserves special attention (see Figure 3.44). The precision graphs show very high precision at the highest levels of forgetfulness, especially with the *training*1 and *training*3 sets, and low recall. So, when we combine the precision and recall metrics we can find that such high precision values are rapidly compensated, thus obtaining the expected bell-shaped graph.

Fallout. Fourth, the fallout of the algorithm remains more or less stable whatever the level of forgetfulness (see Figure 3.45).

NCases Fifth, the NCases is slightly different depending on the initial item set (see Figure 3.46). The set with the lowest NCases is *Training*1 since it has the lowest recall and, therefore, it discovers less information. On the contrary, the set with the highest NCases is *Training*0, which also has the highest number of initial items in the case base. It is important to note that the forgetting mechanism reduces considerably the size of the case base whatever the initial item set.



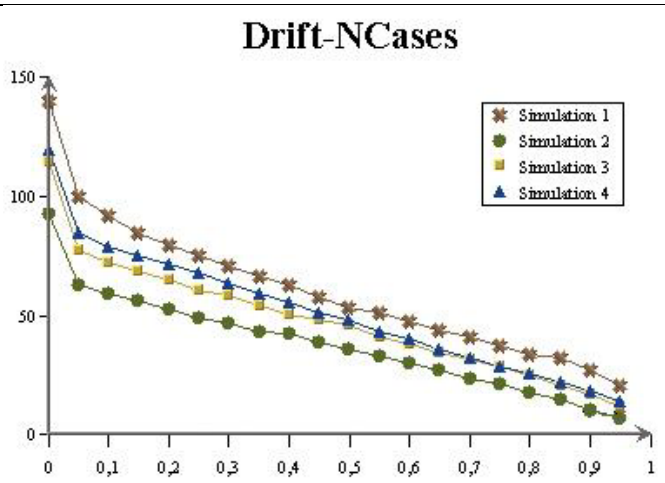
PARAMETERS

CBR/E
6 *months*

Simulation1 - *Training*0
Simulation2 - *Training*1
Simulation3 - *Training*2
Simulation4 - *Training*3

$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.45: Fallout of the system with different initial item sets.



PARAMETERS

CBR/E
6 *months*

Simulation1 - *Training*0
Simulation2 - *Training*1
Simulation3 - *Training*2
Simulation4 - *Training*3

$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.46: NCases of the system with different initial item sets.

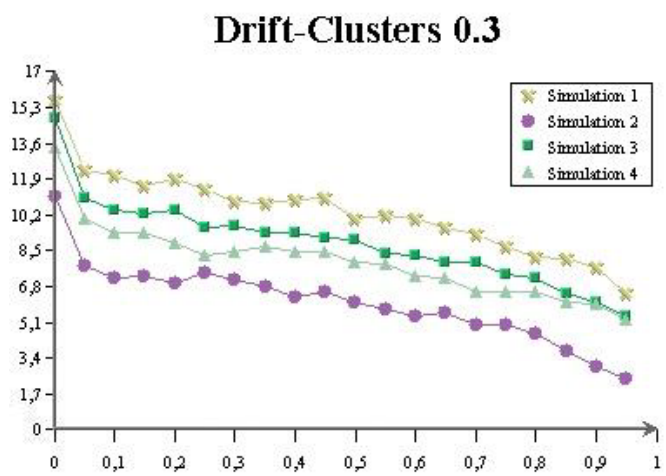


Figure 3.47: Diversity of the system with different initial item sets.

Diversity Sixth, the initial item set strongly affects the loss of diversity in the user profile. Figure 3.47 shows, on the one hand, that when $Training1$ set is used, 55% of the clusters are lost between not forgetting and forgetting a lot ($\xi = 0.7$). $Training3$ set also loses a significative percentage of clusters, 51%. The main reason for these results is that both sets, but especially $Training1$, begin the simulations with extremely low diversity. If the recommendation algorithm forgets too much, is very difficult to increase this level of diversity. However, if the system does not forget at all, the recommendation algorithm remembers the recommended items and can try others, producing an important increase in diversity. On the other hand, when the simulations begin with a diverse initial set of items ($Training2$ and especially $Training0$ set), the increase in diversity obtained without forgetfulness is less important.

Accuracy. Finally, from the point of view of accuracy, $Training1$ and $Training3$ sets perform slightly better than the others (see Figure 3.48). Thus, if the requirement of the application is to obtain the best accuracy, one of these techniques should be selected. In any case, what we really have demonstrated with this graph is that accuracy remains stable whatever the initial item set and whatever the level of forgetfulness.

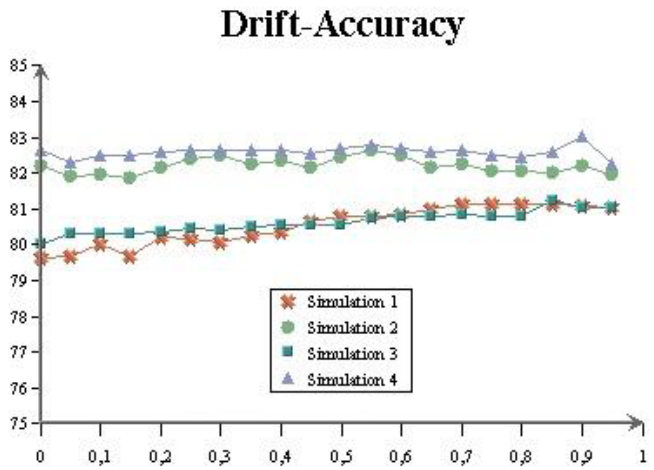


Figure 3.48: Accuracy of the system with different initial item sets.

Summing up, the results indicate the importance of the initial item set for the system when the forgetting mechanism is applied. As observed in our experiments, an initial item set with only a few items suitable for the forgetting mechanism is the simulation that shows the best results. However, since it is very difficult to know which are the suitable items for every algorithm, we believe that the best initial item set is the one containing a large group of items representing the most and the least liked for the user (as these are the most relevant items at the beginning).

Rewarding and Penalising Factors

The drift penalising factor (σ) and the drift rewarding factor (λ) are used to modify the drift attribute of a successful or a failed recommendation respectively. That is, these parameters are used to increase or decrease the relevance of the cases that lead to successful or failed recommendations respectively. A value of $\lambda = 1$ and $\sigma = 1$ means that a successful or failed recommendation does not affect the relevance of the source cases (the retrieved cases from which the recommendation has been computed). A value of $\lambda > 1$ means that the relevance of the source cases are upgraded when a recommendation is successful. A value of $\sigma < 1$ means that a failed recommendation decreases the relevance of the source cases, falling into oblivion. What we want to study with these parameters is, first, their importance in the forgetting mechanism and, then, how big or small to make them. To do so,

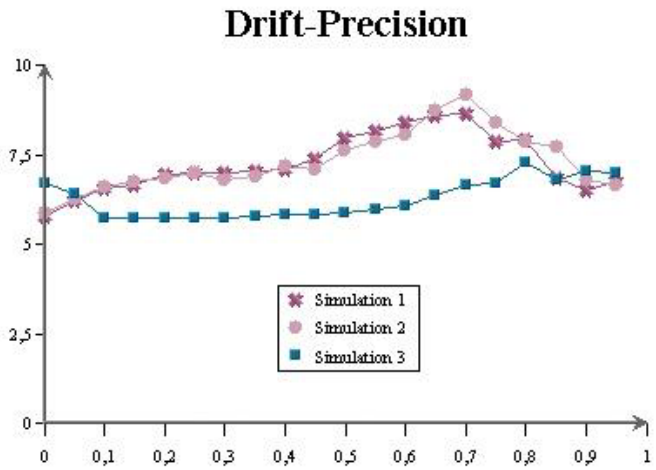
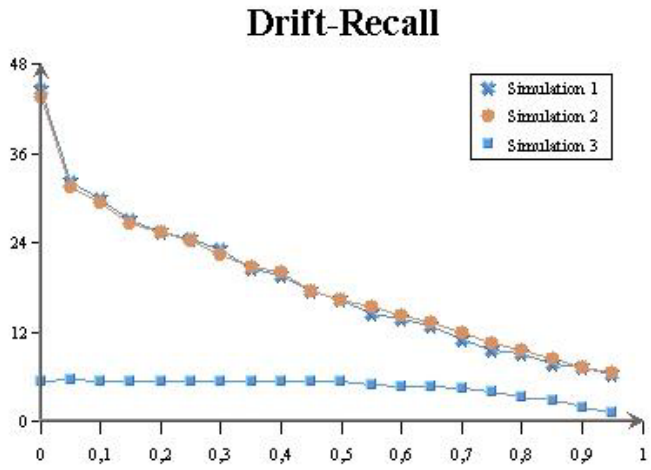


Figure 3.49: Precision of the system with different rewarding/penalasing factors.

we used test values of $\lambda = 1.0$ $\sigma = 1.0$ (no reward/penalty), $\lambda = 1.05$ $\sigma = 0.95$ (a small reward/penalty) and $\lambda = 1.1$ $\sigma = 0.9$ (a larger reward/penalty). The results are presented in Figures 3.49 to 3.54.

Precision. First, Figure 3.49 shows the precision of the recommendation algorithm with the different rewarding and penalising factors. Since the forgetting mechanism is based on knowledge about which are the relevancy and irrelevancy of cases, when the system does not reward or penalise cases ($\lambda = 1.0$ $\sigma = 1.0$), the performance of the system decreases significatively. However, when cases are rewarded and penalised ($\lambda = 1.1$ $\sigma = 0.9$ and $\lambda = 1.05$ $\sigma = 0.95$), the graphs are practically identical.

Recall Second, recall is extremely different when the rewarding and penalising factors are applied, as shown in Figure 3.50. The system does not know the most relevant cases at any one moment and, therefore, always uses the same cases (the ones most liked by the user) as source cases to recommend. Obviously, if the source cases are always the same, the resulting recommendations are also very similar and recall is quite low. However, no difference can be observed between the recall graphs when the relevance is updated.



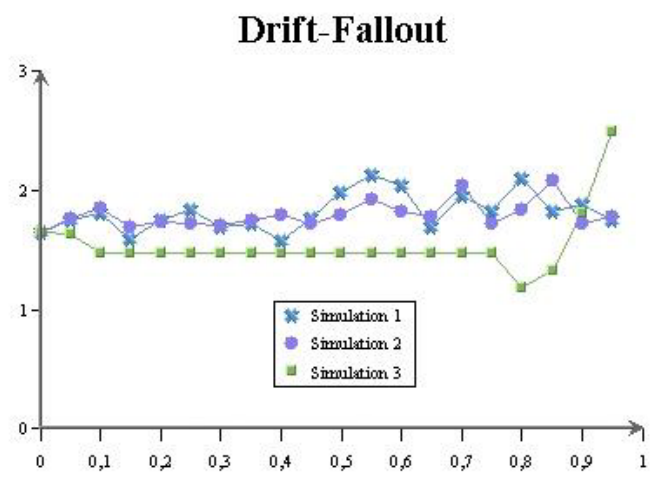
PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$

Simulation1 - $\lambda = 1.1$ $\sigma = 0.9$
Simulation2 - $\lambda = 1.05$ $\sigma = 0.95$
Simulation3 - $\lambda = 1.0$ $\sigma = 1.0$

$\phi = 0.5$ $\gamma = 0.499$

Figure 3.50: Recall of the system with different rewarding/penalasing factors.



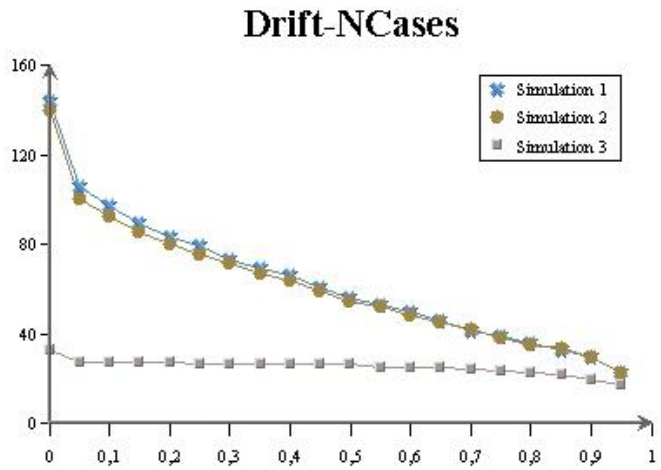
PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$

Simulation1 - $\lambda = 1.1$ $\sigma = 0.9$
Simulation2 - $\lambda = 1.05$ $\sigma = 0.95$
Simulation3 - $\lambda = 1.0$ $\sigma = 1.0$

$\phi = 0.5$ $\gamma = 0.499$

Figure 3.51: Fallout of the system with different rewarding/penalasing factors.



PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$

Simulation1 - $\lambda = 1.1$ $\sigma = 0.9$
Simulation2 - $\lambda = 1.05$ $\sigma = 0.95$
Simulation3 - $\lambda = 1.0$ $\sigma = 1.0$

$\phi = 0.5$ $\gamma = 0.499$

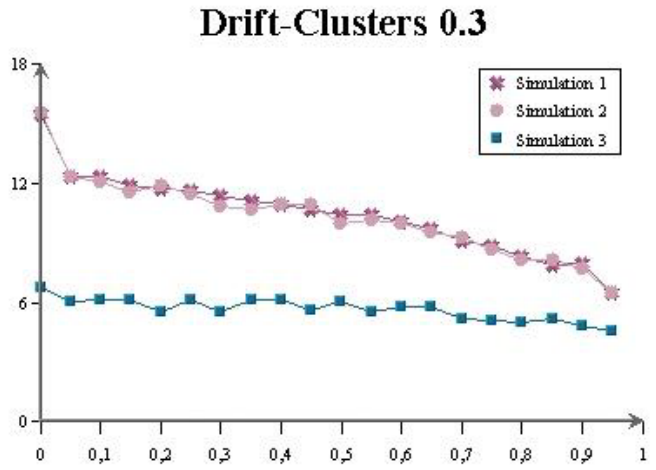
Figure 3.52: NCases of the system with different rewarding/penalasing factors.

Fallout. Third, the fallout graphs are not identical, but they are very similar (see Figure 3.51). If we analyse any level of forgetfulness, the fallout of all three simulations have very similar values.

NCases and Diversity. Fourth, the charts representing the average of cases in the case base and the diversity of the user profile are very similar to the recall graph (see Figure 3.52 and Figure 3.53). Since recall is very low, the NCases and diversity is also very low and, therefore, when the degree of forgetfulness is higher ($\xi = 0.7$), the percentage of cases in the case base and the loss of diversity is very low as well. Once again, when different magnitudes of these reward/penalty parameters are analysed, no significative differences are found.

Accuracy. Finally, the accuracy of the algorithm is affected by the application of these parameters. Like precision, accuracy is significantly lower when cases are not rewarded and penalised. Once again, the accuracy of the system when cases are rewarded and penalised is almost the same whatever the magnitude of these parameters.

Therefore, after analysing the different measures, we can say that a rewarding and a penalising factor which controls the relevance of the cases is essential. With these



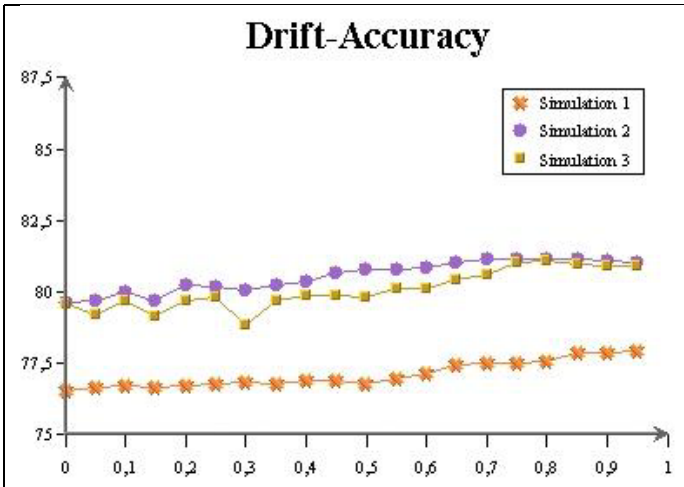
PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$

Simulation1 - $\lambda = 1.1$ $\sigma = 0.9$
Simulation2 - $\lambda = 1.05$ $\sigma = 0.95$
Simulation3 - $\lambda = 1.0$ $\sigma = 1.0$

$\phi = 0.5$ $\gamma = 0.499$

Figure 3.53: Diversity of the system with different rewarding/penalasing factors.



PARAMETERS

CBR/E
6 *months*
$Training0$
$\beta = 0.98$

Simulation1 - $\lambda = 1.1$ $\sigma = 0.9$
Simulation2 - $\lambda = 1.05$ $\sigma = 0.95$
Simulation3 - $\lambda = 1.0$ $\sigma = 1.0$

$\phi = 0.5$ $\gamma = 0.499$

Figure 3.54: Accuracy of the system with different rewarding/penalasing factors.

factors, the system is able to work out which cases should be used in order to find out new successful recommendations and, therefore, the performance of the system is drastically improved. However, the magnitude of these factors is not important, since the performance of the system is the same whatever they are.

Simulation Duration

Another important parameter is the length of the simulation. Simulations are good tools for analysing system performance, but they have some limitations. The fact that we are trying to simulate the development of the user profile over time and that the available information about the user is limited, restricts the time scale of the results that can be calculated. Initially, the system uses the information of the initial profile in order to find out new interesting items. Then, the simulator checks the information available to see whether the item has been evaluated by the user (successful or not). The new item is inserted in the user profile in order to learn about the user preferences and also to adapt the user profile. Then, the system has more information on which to base its recommendations and new items are selected as interesting and recommended to the user. This process is repeated until all the information available has been exhausted and no more evaluations can be performed. Then, since the algorithm cannot find any new recommendations, the same group of items are recommended over and over again. This situation would not happen if there was an interaction with the user, since information from the user would change the user profile. In spite of these limitations imposed by our simulator, we can use it to analyse how many recommendations are required in order to achieve a case base adapted to the user; that is to say, how many simulation cycles (days) the system requires in order to get to know the user interests.

With this aim in mind, we tested four different simulation durations: 1 month, 6 months, 1 year and 2 years. A recommendation cycle is performed every day, so 1 month means that 30 recommendations have been performed. The results of these simulations are presented in Figures 3.55 to 3.60.

Precision and Recall. The precision of the recommendation algorithm is the same for all the time periods we simulated for except 1 month (see Figure 3.55). Most of the simulations discover all the possible items at most within two and a

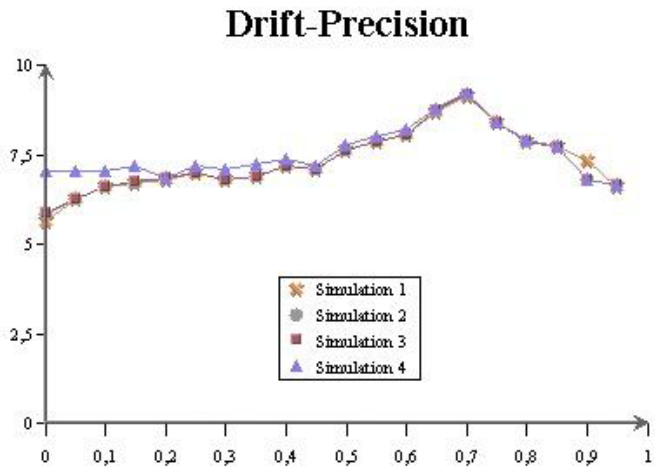


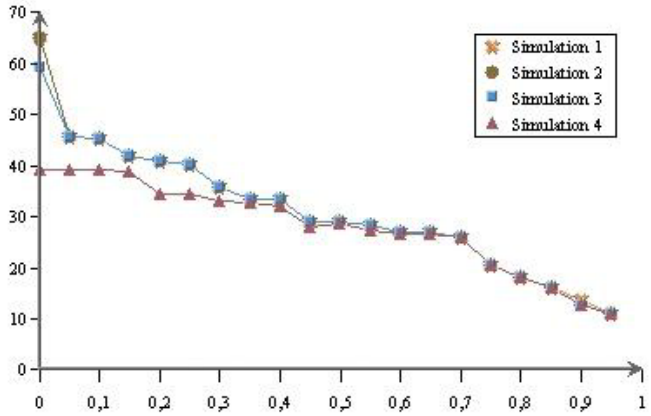
Figure 3.55: Precision of the system with different simulation durations.

half months. However, when we simulated for 1 month, the results are a little bit different since the simulations end in the middle of the discovering process. The recommendation algorithm have not had enough time to recommend all the items that appear to be interesting to the user and, in consequence, the recall of the recommendation algorithm is lower than for the other simulations (see Figure 3.56) and the precision is higher.

Fallout. The fallout graphs are very similar and once again it is demonstrated that the fallout remains stable whatever the parameters and whatever the level of forgetfulness (see Figure 3.57).

NCases. Of course, the NCases is affected by the simulation duration(see Figure 3.58). When the simulation is executed for a month, the discovering process has not finalised and the case base contains less cases than it does for the other simulations.

Diversity. Since, as in the other graphs, the duration of the simulation only affects the very low levels of forgetfulness, diversity presents significative differences only when $\xi = 0.0$ (see Figure 3.59). The simulation has had less time to retain cases and, therefore, diversity is lower. Hence, the loss of diversity is lower (29%) when the simulation is executed for a shorter time.



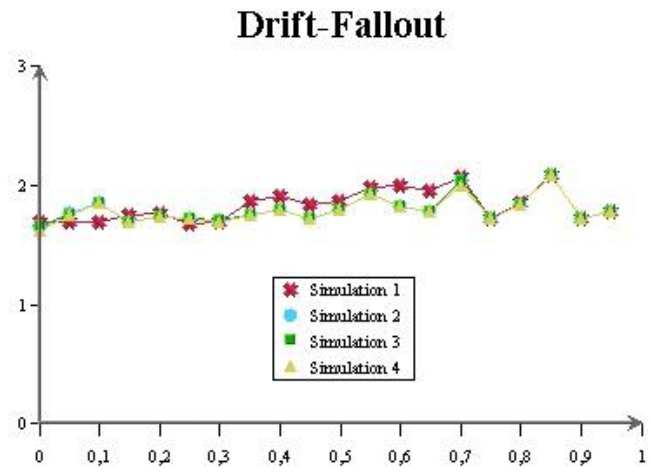
PARAMETERS

CBR/E

Simulation1 - 2 *years*
Simulation2 - 1 *year*
Simulation3 - 6 *months*
Simulation4 - 1 *month*

*Training*0
$\beta = 0.98$
$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.56: Recall of the system with different simulation durations.



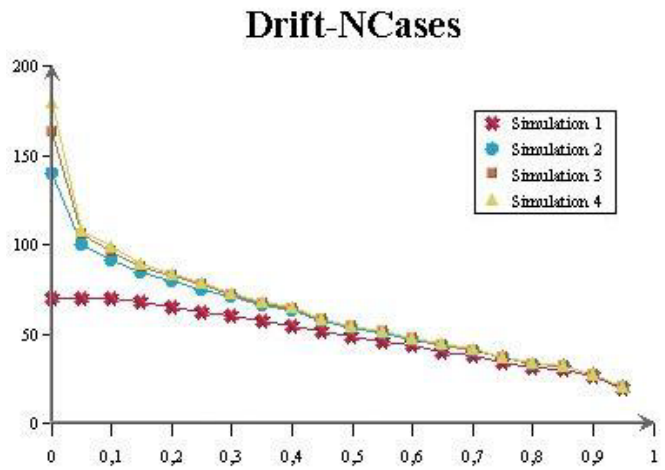
PARAMETERS

CBR/E

Simulation1 - 2 *years*
Simulation2 - 1 *year*
Simulation3 - 6 *months*
Simulation4 - 1 *month*

*Training*0
$\beta = 0.98$
$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.57: Fallout of the system with different simulation durations.



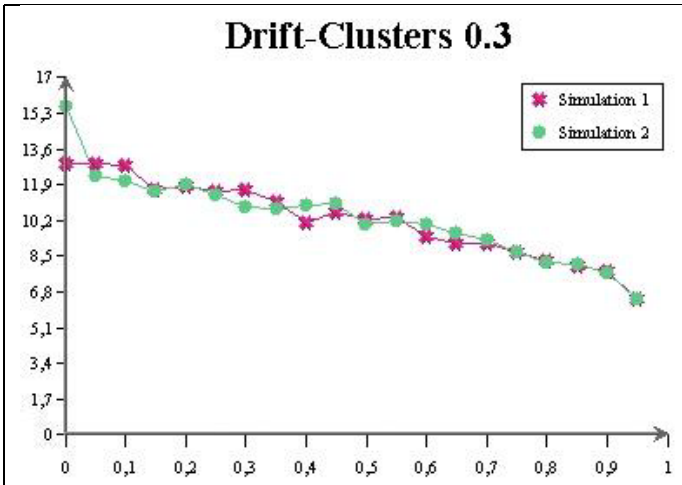
PARAMETERS

CBR/E

Simulation1 - 2 *years*
Simulation2 - 1 *year*
Simulation3 - 6 *months*
Simulation4 - 1 *month*

*Training*0
$\beta = 0.98$
$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.58: NCases of the system with different simulation durations.



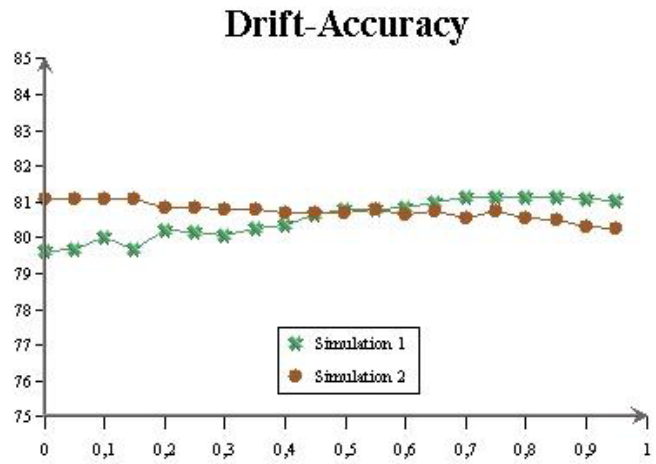
PARAMETERS

CBR/E

Simulation1 - 1 *month*
Simulation2 - 6 *months*

*Training*0
$\beta = 0.98$
$\lambda = 1.05$ $\sigma = 0.95$
$\phi = 0.5$ $\gamma = 0.499$

Figure 3.59: Diversity of the system with different simulation durations.



PARAMETERS

CBR/E

Simulation1 - 6 *months*
Simulation2 - 1 *month*

$Training0$
$\beta = 0.98$
$\lambda = 1.05 \quad \sigma = 0.95$
$\phi = 0.5 \quad \gamma = 0.499$

Figure 3.60: Accuracy of the system with different simulation durations.

Accuracy. Finally, we obtain a better accuracy after 1 month of simulation when the algorithm does not forget at all. However, we do not consider this difference significant since the decrease observed in this simulation is very similar to the increase observed when 6 months are simulated.

Results on simulation duration gives us an idea about the behaviour of the forgetting mechanism over time. Time is necessary to progressively adapt the user profile to the target user. Our experiments have shown that with a static user; that is, a user that does not change his preferences over time, slightly less than a month (30 recommendations) is needed in order to perfectly adapt the user profile and obtain the best performance, since the performance of the system in the proposed level of forgetfulness ($\xi = 0.7$) is exactly the same for 2 years as it is for one month. Experiments of only a few days could be performed in order to find out exactly how many days the forgetting mechanism needs to adapt the user profile, but since this adaptation depends very much on the particular user and the dynamism of their preferences, we can say that, at most, only 1 month is needed for a static user. We think that this result can be generalised to more dynamic users (users that change preferences over time) since typical users do not change their interests every month.

3.7 Chapter Conclusions

In this chapter, we have presented a new approach to recommendation, based on CBR. The main contribution of this approach is the forgetting mechanism which we base on the drift attribute. The drift attribute controls the relevance of each case, taking into account whether the given case leads to successful/failed recommendations and the currency of the user interest in such cases. Thus, the drift attribute lets the recommender system distinguish between relevant and irrelevant cases. Consequently, relevant cases will be more confidently recommended than irrelevant ones. However, the most important advantage of applying the drift attribute is that the utility problem is mitigated. Drift cases are deleted and the number of cases in the case base becomes stable while the performance of the system is maintained and even improved.

The aim of this work is not to present a new recommendation algorithm that performs better than the rest. We simply want to demonstrate that our forgetting mechanism, proposed for case-based profiles, improves the quality of recommendations. Therefore, everybody can implement the drift attribute in their own recommender systems in order to improve their performance.

In order to validate our proposal, we have provided an experimental platform for CBR recommender systems based on the profile discovering procedure. Using this evaluation method, we have implemented a simulator that has demonstrated the outstanding performance of the proposed forgetting mechanism.

The experiments have shown that the forgetting mechanism produces an increase in precision, a decrease in recall and an important reduction in the number of cases in the case base for all the recommendation algorithms, while preserving fallout and accuracy. In particular, the precision of the system is drastically improved with the proposed forgetting mechanism. The best precision is obtained when the recommender system forgets substantially. Forgetting too much is not good, although not forgetting anything is even worse. In other words, from the point of view of precision, forgetting at any level is better than not forgetting at all.

After presenting the results of the drift attribute from a global point of view, we analyse how different recommendation algorithms, different forgetting parameters, different initial item sets and different simulation durations affect the performance of the forgetting mechanism. After an exhaustive analysis, we can conclude:

- The forgetting mechanism improves the quality of recommendations whatever the recommendation algorithm, provided that they take into account the relevance of the user's interest.
- Depending on the objectives of the recommender system, we can choose between a set of algorithms with a low recall and a very high precision and a set of algorithms with a higher recall but a lower precision.
- Algorithms that incorporate implicit information (captured from the interaction between the user and the system) from the recommendation process do not provide significative improvements to the system performance.
- The speed of forgetting is strongly correlated with the forgetfulness threshold. Therefore, choosing a good relationship between these parameters is very important.
- The behaviour of the forgetting mechanism does not depend on the user's subjective assessments that define what is a success and what is a failed recommendation.
- The performance of the system depends strongly on the synergy between the recommendation algorithm and the initial item set of items forming the initial user profile. The best initial profile has to contain the most liked items, since they are the most relevant at the beginning.
- A rewarding and a penalising function that controls the relevance of cases in the case base is essential. However, the magnitude of these parameters is not important.
- Less than 30 recommendations are necessary in order to perfectly adapt the case-based profile to the user.

Of course, there are more combinations of parameters but we think that the experimental work presented in this chapter is exhaustive enough to prove the validity of the forgetting mechanism approach we propose for CBR recommender systems.

Chapter 4

Collaborative Recommender Agents

Recommender systems sharply improve the quality of their results when information about other users is utilised when recommending to a given user. The collaborative filtering method has shown important results in this sense, although this method requires the revelation of personal information about the users. In order to maintain the privacy of the users' personal data, we propose a new mechanism of collaboration based on intelligent agents. Agents encapsulate the user profile and are in charge of recommending interesting items to the user. Collaboration is achieved by means of two methods based on the real world: the opinion-based filtering method and the collaborative filtering method through trust. Both are based on a social model of trust that provides the agents with a tool to decide with whom to collaborate. In this chapter, we also propose an evaluation procedure for collaborative recommender agents. The results show that our proposal improves the performance of a typical collaborative recommender system while preserving privacy.

4.1 Introduction

In the real world, society in general, but especially our friends, help us to find new and amazing things. Often our friends advise us an interesting product, movie, book or restaurant, collaborating with us in a selection process. Being aware of the relevance of collaboration in the real world, researchers have focussed on the

development of recommender systems which can recommend items to a user based on information from other users.

Particularly, the collaborative filtering method has proved to be an useful method to take advantage of the collaborative world [Resnick 94, Shardanand 95, Breese 98, Herlocker 99], especially when combined with other technologies in a hybrid approach [Balabanovic 97, Pazzani 99, Good 99]. This method recommends new items based on the similarity between user profiles. In particular, given a user profile, a set of similar profiles is found comparing the information contained in them (typically item evaluations). Then, the best evaluated items of the other profiles that the given user has not evaluated are recommended.

However, collaborative filtering systems have an important drawback: the lack of privacy of user's personal data. The huge amount of user profiles centralised in a server are a valuable data for marketers who want to invade people's privacy for profit. Who is interested in what is indeed useful information for enterprises in designing their marketing campaigns. Thus, user's personal data can be used as a powerful tool for good (collaboration) or bad (commercial benefits).

Therefore, recommender systems have to ensure the privacy of the people behind user profiles and the personal data contained in such profiles (for example, demographic information as names, age or profession, and information about interests such as purchased products, visited restaurants or preferred books). Recommender systems can protect user's personal data on two levels. A first level ensuring privacy against people outside the recommender system and a second and deeper level protecting personal information against other users within the system.

Increasingly, governments require enterprises to protect their systems at the first level by law. The hardest problem arises when we want to protect systems at the second level. In this direction, the scientific community is dedicating a great deal of effort to the so-called *recommender agents* [Klusch 01]. Recommender agents are like personal assistants who recommend interesting items to their users based on past experiences. Recommender agents belong to a multi-agent system (MAS) where agents with different functions coexist and collaborate in order to achieve their purposes. The most interesting property of such agents is that they encapsulate the user profile. The agent's knowledge is not accessible by other agents nor by other users. Moreover, since recommender agents are typically installed in the user's

computer, all information relating to the user is kept in private in a local machine instead of a centralised server. Then, either the user or their own agents decide what information to share and how. Anyway, in both cases the privacy of the user's personal information is protected. Finally, the anonymity of the users is protected by the MAS infrastructure, which impedes the agents of the system from knowing who the person behind a recommender agent is.

Privacy is achieved, but not without cost. If we want to take advantage of the collaborative world, we have to share information about our interests and preferences in order to find similar users who can help us. However, broadcasting personal information raises privacy concerns. People want their privacy as well as the benefits of personalised information, thus, there is a serious trade-off between privacy and collaboration.

If recommender agents protect the user's information, the collaborative filtering method cannot be applied since a direct comparison among profiles is not possible. The solution we propose is a new information filtering method: the opinion-based filtering method. Its main idea is to recommend new items taking into account the best friends' opinion. That is, recommender agents will recommend a new item that seems interesting to their users provided that their best friends give them good opinions on it. Giving an opinion implies showing personal interests. However, if an opinion is a value between 0 and 1 indicating from a total affinity to the item (1) to a total dislike (0), no specific personal information is revealed. For example, giving a good opinion on a certain product does not indicate that the user buys it. Maybe the user likes this kind of product but has never bought it for some reason. Thereby, an opinion on a certain item is a value that aggregates all the information contained in the user profile regarding the item.

Only the opinions provided by best friends are taken into account in the recommendation process. In order to know which agents are good or bad friends, recommender agents consider other agents as personal entities on whom they can rely or not. Reliability is expressed through a trust value with which each agent labels its neighbours. If a recommender agent A labels another agent B with a high trust value, B is a good friend of A. Initially, trust is computed as a similarity between opinions. Agents exchange and compare opinions in order to find similar agents. The trust values of the agents are then modified depending on the profitableness of the opinions and advice provided. Thus, the opinion-based filtering

method provides recommender agents with a mechanism that allows them to look for similar agents who can offer them opinions and advice, assuming that similar agents have similar opinions about items.

We can now compare opinions by means of the opinion-based filtering method and find out similar profiles; that is, friends. Therefore, thanks to the trust values obtained through the opinion-based filtering method, an improved version of the collaborative filtering method can be applied. In this new approach, recommender agents only ask for recommendations to the agents with a high trust value.

In this chapter, we introduce this new approach to collaborative recommender agents. These agents encapsulate user information, ensuring privacy. Then, they use the opinion-filtering method to proactively ask other agents for their opinion in a "lack of information situation" instead of remaining passive or providing either a negative or an empty answer to the user. Finally, the agent receives new recommendations from other agents in a new collaborative environment based on trust. Thus, our social model exploits interactiveness while preserving privacy.

This chapter is structured as follows. The following section contextualises our proposal within the current state-of-the-art. Sections 4.3.4 and 4.4 present the new information filtering methods proposed (opinion-based and collaborative trust-based, correspondingly). Section 4.5 introduces the formal social model of our approach to trust for recommender systems. Then, how the proposed methods has been evaluated and experimental results are shown in sections 4.6 and 4.7 correspondingly. Finally, in section 4.8 we provide some conclusions.

4.2 Related Work

Regarding privacy, a great deal of work has recently been done in order to protect personal data in collaborative filtering systems. The assignation of pseudonyms to users, for example in GroupLens [Resnick 94], has proved insufficient, since users can be tracked. Bowbrick proposed the improvement of privacy protection by means of multiple identities [Bowbrick 00]. Users are permitted to create and manage separate identities, which may or may not reference the same user profile. They now have the choice as to which identity to present to which site. We believe that in this proposal too much work is delegated to the user.

Many researchers bet on the generation of communities of agents with similar interests, where recommendations are provided from the community while the personal information of each particular member is camouflaged. A great deal of effort has been dedicated to the field of artificial immune systems (AIS) in order to generate these communities in an anonymous way [Dickinson 03, Cayzer 02]. The natural immune system is an adaptive learning system inspired by biology that employs many parallel and complementary mechanisms for defense against foreign pathogens. These ideas are very similar in spirit to the construction algorithm proposed by Delgado [Delgado 01], and to the ideas of a distributed query, as used in Gnutella [Gnutella 03]. Canny introduces an encryption mechanism when generating communities [Canny 02a, Canny 02b]. First of all, users decide what they want to share. We claim that agents have to do this work, since we have designed them in order to liberate the user of such annoying tasks. Then, the protocol proposed is used in order to encrypt the shared information before aggregating it with the information from other users in the community. The resulting information is published with the aim of collaboration. We believe that generalisation at this level could strongly affect the performance of the recommender agent. Our proposal also wants to generalise, but to a lesser degree.

Regarding trust, there are very few approaches to trust in the collaborative world applied to the information filtering field. Knowledge Pump is an information technology system for connecting and supporting electronic repositories and networked communities [Glance 98]. Glance et al. introduce a technique that they call Community-Centred Collaborative Filtering (CCCF). In CCCF, the collaborative filter is bootstrapped by a partial view of the social network constructed from a user-input list of "advisors" (people whose opinions users particularly trust). The set of advisors is generated through statistical algorithms that mine the usage data automatically. The collaborative filter gives a higher weight to the opinions of his/her most trusted contacts when predicting the user's opinion on items. The main difference from our model is the computation of the trust value since Glance bases it on a person-person correlation. So transparency of user data is required through agents, while in our system privacy prevails.

In other fields, such as electronic commerce, we can find other trust models that fit the particularities of the domains. For example, Schillo et al. present a formalisation and an algorithm for trust so that agents can autonomously deal

with deception and identify trustworthy parties in open systems [Schillo 00]. They demonstrate with results that their approach helps each single agent to establish a model of trustworthiness of other agents. With only a few iterations, agents learn whom to trust and whom to exclude from future interactions. They also show that agents form groups and interact to profit from mutual support. Previously, Schillo had implemented a relevant computational method in the Social Interaction FrameWork (SIF) [Schillo 99] in which an agent evaluated the reputation of another agent on the basis of direct observation and through other witnesses. The idea of using the opinion of other agents to build a reputation is also applied by Yu and Singh [Yu 00]. Their agents build and manage trust representations not only taking into account previous experiences of their users, but also communicating with other agents (belonging to other users). They aim at avoiding interaction with undesirable participants and formalising the generation and propagation of the reputation in electronic communities.

4.3 The Opinion-Based Filtering Method

The main idea of the opinion-based information filtering method is to consider other agents as personal entities on whom you can rely or not. Reliability is expressed through a trust value with which each agent labels its neighbours. The trust value is initially computed through interaction, following the proactive *playing agents* procedure explained in section 4.5.2.

Once the agent has a set of *friends*, it can use them to filter information. When agents are not sure about a recommendation or discover a new item, they ask their reliable agents for an opinion and use their trust values to decide whether the item is interesting to the user or not. Once the agent has the opinion of the other agents, a consensus is achieved through the use of an aggregation measure. The result of the consensus provides a confidence value upon which the agent can decide on the convenience of recommending an item to the user or not. Thus, instead of using the best friends opinions directly as a recommendation, the agent includes it in its own reasoning and combines it with other agents' opinions in order to decide whether to recommend a given item. We call this new process of filtering information based on agents opinions the opinion-based information filtering method.

Thus, the opinion-based filtering method is based on the exchange of opinions among reliable recommender agents. But what is an opinion? An opinion on an item is a value representing the interest the agent thinks his/her user has in that item. Opinions are computed in two different ways depending on whether the user has had any experience on it or not; that is, if the user has any evaluation of it in the user profile.

We must now define how we represent the user profile and then explain how we calculate, from the user profile, an opinion on either an item evaluated by the user or a non-evaluated item. Finally, we describe how the opinion-based filtering method is used in order to recommend new items to the user.

4.3.1 User Profile Representation

We consider a user profile representation as a case base containing past experiences on certain items according to the CBR approach presented in the previous chapter (see section 3.3.1).

For example, in the restaurant domain, items and interests are represented as:

$$A_t = \{name, address, phone\ number, cuisine, approximate\ price, capacity, web\ page\}$$

$$Int^e = \{quality/price\ relation, quantity\ of\ food, table-waiting\ efficiency\}$$

$$Int^i = \{web\ page\ visits,\ recommended\ times,\ retrieved\ times\}$$

A single experience of the user in the Mallorca restaurant is:

$$E =< \{"Mallorca\ Restaurant",\ "2228\ East\ Carson\ St,\ Pittsburgh, PA",\ "(412)4881818", "Spanish", "\$70", 300,\ "www.mallorcarestaurant.com"\},\ \{0.83, 0.76, 0.91\},\ \{2, 21, 12\},\ 0.83 >$$

The experience of agent a_i in item p_j is $E_{i,j}$, and the set of all possible experiences is denoted as $\mathcal{E}$. The set of past experiences constituting the user profile is denoted as $\mathcal{E}_i$, where $\mathcal{E}_i \subset \mathcal{E}$.

In an open environment such as Internet, recommender agents collaborate in what is called a multi-agent system (MAS). The function of these agents is to recommend new items to their users. In order to achieve this purpose, recommender agents collaborate exchanging information with each other. Hence, a list of agents which the agent trusts is also incorporated in the user profile representation.

Given a set of agents

$$A = \{a_1, a_2, \ldots, a_r\},$$

each agent a_i has a list of contact neighbourhood agents on which it relies:

$$C_i = \{(a_{i_1}, t_{i,i_1}), (a_{i_2}, t_{i,i_2}), \ldots, (a_{i_n}, t_{i,i_k})\}$$

where $a_{i_j} \in A$ and t_{i,i_j} is a numerical value between [0,1] that represents the trust value that agent a_i has in agent a_{i_j}. Like the set of experiences, the contact list has to be initialised in order to start the collaboration as soon as possible. The initial contact list is generated by means of the "playing agents" procedure explained in section 4.5.2.

Therefore, we represent the user profile of agent a_i with the past experiences and the set of selected agents who agent a_i trusts:

$$Prof_i =< \mathcal{E}_i, C_i >$$

4.3.2 Opinion On an Evaluated Item

Instead of revealing all the user's personal information regarding an evaluated item, an opinion is a value that aggregates all the interest attributes of the experience in a given item. Exchanging all the information about the users' interests in the evaluated item would violate their privacy. However, a simple value representing their opinion hides detailed information and preserves personal data.

Thus, an opinion consists of a quantitative value, between 0 and 1, which represents the degree of interest the agent thinks the user has in the item ranging from a total affinity (1) to a total dislike (0). This value is calculated aggregating all the information about the interests of the user in the given item. Thus, the opinion $V_{i,j}$ of an agent a_i in a item p_j is calculated following equation 3.1.

Applied to the previous example on the experience of the user in the Mallorca restaurant, we have the following values:

- First, for example, the weights of the explicit interest attributes selected by the user are:

Explicit Attributes	Weights
quality/price relation	0.6
quantity of food	0.1
table-waiting efficiency	0.3

Therefore, $f_j^e = 0.6 * 0.91 + 0.1 * 0.83 + 0.3 * 0.76 = 0.86$

- Second, the function we have selected to aggregate the implicit attributes proposed is the following:

$$f^i = 0.35 * \frac{2 * atan(v)}{\pi} + 0.45 * \frac{2 * atan(\frac{1}{r})}{\pi} + 0.2 * \frac{2 * atan(c)}{\pi} \qquad (4.1)$$

where v is the number of visits to the restaurant web page, r is the number of times the restaurant has been recommended to the user and c is the number of times the restaurant has appeared as a result of a query performed by the user (retrieved times).

Therefore, $f^i = 0.35 * \frac{2*atan(2)}{\pi} + 0.45 * \frac{2*atan(\frac{1}{20})}{\pi} + 0.2 * \frac{2*atan(12)}{\pi} = 0.44$

- Then, the application of the g function to aggregate explicit and implicit general evaluations is calculated with $\rho_e = 0.7$ and $\rho_i = 0.3$:

$$g(f_j^e, f_j^i) = 0.7 * 0.86 + 0.3 * 0.44 = 0.73$$

- Finally, the interest value representing the user's opinion on the item is computed taking into account the relevance of the item:

$$V_{i,j} = 0.83 * 0.73 = 0.61$$

4.3.3 Opinion On a Non-Evaluated Item

An opinion on a non-evaluated item is calculated from other evaluated items with similar item attributes. Since the user has had no experience with the item, experiences with similar items are used. For example, if a recommender agent is enquired about a non-evaluated restaurant and the user has had good evaluations of several other restaurants with similar cuisines and prices, the recommender agent can suppose the user will have a good opinion on the new restaurant.

In order to calculate the opinion on an enquired item r that the user has not evaluated and according to the CBR approach presented in section 3.3, the following steps are performed:

1. The similarity Sim between the enquired item r and all the items contained in the case base is computed through similarity functions (see section 3.3.2).
2. A set of best matches is chosen. In our implementation, we select the x best cases retrieved which exceed a minimum selection threshold.
3. The item interest value V_i of each selected item i is computed using equation 3.1.
4. The interest confidence value I of the enquired item r is computed as a weighted ponderation of the item interest values of each similar item (see equation 3.4).
5. The interest confidence value I_r is the opinion provided.

It is important to note that giving opinions using this CBR approach reinforces the privacy concept. Recommender agents show user interests without revealing concrete information. For example, a good opinion on a restaurant means that the recommender agent thinks the user could be interested in it, perhaps because the user has a good evaluation of it or the user profile shows a liking for similar restaurants.

4.3.4 Recommendations Based on the Opinion-Based Filtering Method

A hybrid approach between content-based and opinion-based filtering methods is used in order to decide which items to recommend to the user. Recommender agents can receive a new item from their environment, or they can also proactively look for new items (for example, asking an agent who provides information about items). When an agent discovers a new item, p_{new}, a content-based filtering method is used in order to decide whether it is interesting for the user. First of all, the agent's opinion on the new item is calculated following the CBR approach proposed in section 4.3.3.

Thus, based on the experiences with similar items, the agent generates an own opinion on the new item. Then:

- If the own opinion on the new item is over a top doubt threshold τ^+, the agent recommends it to the user.
- If the own opinion is under a bottom doubt threshold τ^-,the agent does not recommend the new item to the user assuming that the user has no interest in it.
- If the own opinion is confined within the doubt thresholds (τ^-, τ^+), the agent turns to the opinion-based filtering method in order to decide whether to recommend the new item.

The opinion-based filtering method consists of the following steps:

1. Ask the agents contained in the contact list for their opinion on item p_{new}. For each enquired agent a_{e_i} an item interest value $V_{e_i,new}$ is calculated following equation 3.1.

agent	opinion
a_{e_1}	$V_{e_1,new}$
a_{e_2}	$V_{e_2,new}$
$\vdots$	$\vdots$
a_{e_n}	$V_{e_n,new}$

Table 4.1: Opinion of the agents in the contact list.

2. Compute a global value for the new item, r_{new}, based on the opinion of the queried agents with a trust value over the trust threshold ω. Note that all the agents in the contact list are asked for their opinion, although only the opinions of the friends the agent most trusts are taken into account. It is important to know the opinions of all the agents in the contact list in order to update their trust values (see section 4.5.3). Thus, only best friend's opinions are aggregated. Since we are dealing with several sources of information, an appropriate combination function is the weighted average (WA) where weights are the trust values of the agents. So,

$$r_{new} = \frac{\sum_i^{|C_q|} t_{q,i} * V_{e_i,new}}{\sum_i^{|C_q|} t_{q,i}} \tag{4.2}$$

where $t_{q,i}$ is the trust value agent a_q has in the queried agent a_{e_i}; and $|C_q|$ is the cardinal significance of the contact list of the querying agent a_q.

3. If r_{new} goes above the τ^+ threshold, then the new item is recommended to the user.

4.4 The Collaborative Filtering Method Through Trust

When a recommender system is implemented as a distributed world of recommender agents, the typical collaborative filtering method cannot be applied. The fact that recommender agents encapsulate user profiles makes impossible a direct comparison among them; the basis of the collaborative filtering method.

However, the opinion-based filtering method allows recommender agents to interact and, therefore, compare their interests through opinions and find similar agents who can give them advice. Thus, a new collaborative filtering approach arises.

A typical collaborative filtering method recommends to the user the best evaluated items from similar profiles. In our method, a recommender agent asks his/her reliable agents for a set of items most preferred by their users, assuming that his/her user would prefer them as well. This is what we call "ask for advice". We now describe in more detail how advice is provided and how recommendations are made using the collaborative filtering method through trust.

4.4.1 Advice

When an agent a_o trusts in another agent a_i, a_o can ask a_i for advice. In particular, a_o asks a_i for k recommendations. Then, a_i looks for a set of items, P_k, where P_k contains the k most preferred items by the a_i's user.

Given

$$\mathcal{E}_i = \{E_{i,j_1}, ..., E_{i,j_n}\}$$

as the set of items conforming to the user profile that agent a_i encapsulates, where E_{i,j_m} is the experience of a_i in item p_{j_m}, and n is the number of experiences contained in the user profile.

Then, a_i evaluates the interest confidence values, V_{i,j_m}, of all the items p_{j_m} contained in $\mathcal{E}_i$ by means of Equation 3.1. Thus,

$$\mathcal{E}_i = \{(E_{i,j_1}, V_{i,j_1}), ..., (E_{i,j_n}, V_{i,j_n})\}$$

where V_{i,j_m} represents the interest a_i thinks his/her agent has in item p_{j_m}. Finally, the k items with the highest V_{i,j_m} are sent to a_o.

4.4.2 Recommendations Based on the Collaborative Filtering Method Through Trust

Taking advantage of the trust values with which agents label their neighbours, an improved version of the collaborative filtering method is used in order to obtain new recommendations. In particular, the following steps are performed:

1. Recommender agents ask their best friends (only the agents contained in the contact list with a trust value over ω) for the preferred restaurants of their users. The enquired agents respond with a set of items and their opinion on them.

agent	recommendations
a_{e_1}	$(p_{e_1,1},\ V_{e_1,1}), ..., (p_{e_1,k},\ V_{e_1,k})$
a_{e_2}	$(p_{e_2,1},\ V_{e_2,1}), ..., (p_{e_2,k},\ V_{e_2,k})$
$\vdots$	$\vdots$
a_{e_n}	$(p_{e_n,1},\ V_{e_n,1}), ..., (p_{e_n,k},\ V_{e_n,k})$

Table 4.2: Collaborative recommendations.

2. A confidence value for each advised item is computed by multiplying the friend's opinion V_{e_i} with its trust value $t_{q,i}$. If the same item is recommended by several friends, the confidence value is computed by aggregating the different opinions following equation 4.2, where $|C_q|$ is the number of friends who recommend the given item.

3. Finally, a list of advised items with a confidence value is obtained. The item with the highest confidence value is recommended to the user.

It is important to note that when either opinion-based or collaborative filtering is applied, if the enquired agents provide the interest values of the item; that is, $int^e_1, ...int^e_m$, and $int^i_1, ...int^i_l$, instead of an aggregated value, V_i, the information gathered by the querying agent will be fuller and a more accurate decision can be made. For example, we can use Multicriteria Decision Making techniques (MCDM, [Valls 00]) based on the preferences of the querying agent. However, such information can be considered confidential in some environments. Therefore, in our approach privacy prevails over accuracy.

4.5 Social Trust Model for Recommender Agents

Recommender agents are used to assist users by filtering information. Just as in the real world where people ask their friends for advice on interesting items, an agent should be able to ask for opinions only from reliable agents. Marsh proposes the concept of trust to make our agents less vulnerable to others [March 94]. Trust is basic in any kind of action in an uncertain world; in particular it is crucial in any form of collaboration with other autonomous agents [Castelfranchi 01]. There is no standard definition for trust [Gambetta 90, Castelfranchi 98]. Elofson gives a definition closer to our approach [Elofson 98]. He claims that observations are important for trust, and he defines trust as:

"Trust is the outcome of observations leading to the belief that the actions of another may be relied upon, without explicit guarantee, to achieve a goal in a risky situation"

Elofson notes that trust can be developed over time as the outcome of a series of confirming observations (also called the dynamics of trust). From his experimental

work, Elofson concludes that information regarding the reasoning process of an agent, more than the actual conclusions of that agent, affect the trust in those conclusions.

Trust is formed and updated over time through direct interactions or through information provided by other members of society about experiences they have had. Each event that can influence the degree of trust is interpreted by the agent either as a negative or a positive experience. If the event is interpreted as a negative experience the agent will loose his trust to some degree and if it is interpreted to be positive, the agent will gain trust to some degree. The degree to which trust changes depends on the trust model used by the agent. This implies that the trusting agent carries out a form of continual verification and validation of the subject of trust over time.

Some efforts have been made in the study of social models of trust in market environments [Sabater 00], where several agents compete for their individual profit as well as in other environments where agents need to delegate actions to other agents [Castelfranchi 98]. What we propose is a social model of trust in the information filtering environment, providing recommender agents with a technology to make them less vulnerable to others' opinions. Next, we describe the model by means of the following sections: trust representation, initial trust generation and trust adaptation.

4.5.1 Trust Representation

The trust value agent a_i has in another agent a_o, $t_{i,o}$, is represented with a real value between 0 and 1:

$$t_{i,o} \in [0 - 1]$$

where 1 represents that a_i blindly relies on a_o and 0 that a_i totally mistrusts a_o.

4.5.2 Initial Trust Computation

In order to start the collaboration, recommender agents need to fill in the initial contact list. Thus, reliable agents have to be found in the MAS. We assume that

there is a service in the MAS that provides a list of currently available agents. This assumption is reasonable taking into account that most of the MAS platforms currently available are FIPA compliant [FIPA 03] and provide such service. Then, for each agent, an initial trust value is computed by means of a procedure that we have called *playing agents*. This procedure is based on Steels work [Steels 97] and consists of the following steps:

1. The querying agent selects a set of available agents (enquired agents) in order to compute their initial trust.
2. The querying agent asks an agent in the set for their opinion on one or several items in the initial item set.
3. The opinions of the other agent are compared with the opinions of the querying agent resulting in a similarity value.
4. Steps 2 and 3 are repeated for all the agents in the set of enquired agents.
5. Only agents with a similarity value over a given threshold (ω) are kept in the initial contact list. The similarity value will be the initial trust value.

The first step consists of the selection of a set of recommender agents to contact. We must note that the number of agents in the collaborative world is a matter of constraint. That is, it will be very time-costly if any agent, in order to build a contact list, starts a playing agents procedure with all the available agents in the world. For example, in a platform where agents recommend restaurants in Girona, up to 75.000 agents, one for each citizen, could be considered in the playing agents procedure. To reduce the number of agents to be queried, in each playing agents execution only a random subset of available agents is considered.

Once a set of agents has been selected, the querying agent asks them for their opinions on a set of items from the initial item set; that is, the playing item set (step 2). For example, the best and the worst evaluated item. We can apply this procedure because all the experiences contained in the initial profile have been generated from the same initial item set.

Thus, the current querying agent, a_q, gathers a total of $|P^t|$ opinions from each enquired agent a_{e_i}, one for each item in the enquired set (see table 4.3).

agent	p_1	p_2	...	$p_{\|P_t\|}$
a_{e_1}	$V_{e_1,1}$	$V_{e_1,2}$	...	$V_{e_1,\|P_t\|}$
a_{e_2}	$V_{e_2,1}$	$V_{e_2,2}$	...	$V_{e_2,\|P_t\|}$
$\vdots$				
a_{e_n}	$V_{e_n,1}$	$V_{e_n,2}$	...	$V_{e_n,\|P_t\|}$

Table 4.3: Interest values gathered by the querying agent.

Then, the third step consists of calculating an initial trust value based on the similarity between the opinions of the querying and the enquired agents. The trust that agent a_q has in agent a_e, noted as $t_{q,e}$ is computed as follows:

$$t_{q,e} = \frac{\sum_{i=1}^{|P^t|} \delta_{q_i}(1 - |V_{q,i} - V_{e,i}|)}{\sum_{i=1}^{|P^t|} \delta_{q_i}} \tag{4.3}$$

where $V_{q,i}$ and $V_{e,i}$ are the opinions of a_q and a_e on item p_i correspondingly, and δ_{q_i} is a temporal parameter in [0,1] that indicates the relevance agent a_q gives to item p_i. This parameter is called the drift attribute and is part of the forgetting mechanism explained in chapter 3. This function computes a nearest neighbour similarity between the opinions of both agents, weighted by the relevance of the items according to a_q's interests (the querying agent). The result of the function is a normalised value in [0,1].

Finally, once the querying agent knows the similarity (initial trust) between their opinions and the opinions of the other agents, the initial contact list is generated. This is achieved by means of a trust threshold (ω): only the agents with an initial trust over ω are kept in the initial contact list. Thus, initially agents are not considered reliable either because of their honesty or their trustworthy information but because of similar preferences, interest or styles.

However, recommender agents do not only look for reliable agents at the beginning. Agents should try to make new friends over time. This is a very important issue since, at the beginning, the user profile contains only a little information about the user due to the lack of interaction. As time passes, the user profile better defines what the user is interested in and, therefore, the playing agents procedure is more efficient when looking for reliable friends. Moreover, taking into account that user's interests change in the course of time, having a good friend today does not

mean that I will have the same friend in two years. And viceversa, having different interests today does not mean that we cannot be good friends at a future time.

Be aware that thanks to the CBR approach of the opinion-based filtering method (see section 4.3.3), the sparsity problem seen in section 2.4.1 is solved. After a period of adaptation to the user, when the recommender agent looks for new reliable agents, the coverage of evaluated items becomes very sparse, having many user profiles with different items evaluated. Therefore, only a very few items can be compared in order to find similarities. With our proposal, when a recommender agent asks another agent for his/her opinion on a non-evaluated item, an opinion is generated through the CBR approach proposed in 4.3.3.

Hence, the playing agents procedure is repeated periodically in order to add new agents to the contact list according to the evolution of the user interests. Moreover, the trust value of each agent is updated as a result of a recommendation, as explained in section 4.5.3. In this sense, acquaintance among agents is improved over time.

4.5.3 Trust Adaptation

In order to adapt the contact list, recommender agents need relevant information regarding feedback on recommendations given to the user. If agents provide recommendations based on the opinions or advice of their reliable agents, the trust in these agents should be updated according to the outcomes. The most common way to obtain relevant feedback from the user is by means of the information given explicitly by the user, although the information observed implicitly from the user's interaction with the agent can be also used. In our model, this relevant feedback is captured and kept in the Int^e and Int^i sets, included in each experience of the user profile (see section 3.3.1).

First of all, we need to consider if the recommendation has been successful or not. According to the relevant feedback on the item, the *real* interest of the user in the item V_{real} can be computed following equation 3.1. Since we know the individual opinion of the agents in the contact list (see table 4.1), we are able to analyse which agents agree with the real interest of the user and which agents do not. Then, the trust value of agents should be updated accordingly to their difference with the real interest shown by the user.

Thus, for every agent a_{e_i} in the contact list of agent a_q, its trust value $t_{q,i}$ is updated as follows:

$$t_{q,i} = \varphi * t_{q,i} + (1 - \varphi) * (1 - |V_{real} - V_{e_i,new}|) \tag{4.4}$$

where φ is the trust modifying factor that manages the evolution dynamics of trust. This function was proposed by Jonker and Treur in [Jonker 99]. For a value of $\varphi = 0.8$ we get a slow positive, fast negative dynamics; and for a value of $\varphi = 0.5$ we get a slow negative, fast positive dynamics. Obviously, a slow positive, fast negative dynamics is more appropriate for critical domains where a negative experience is strongly penalised.

However, if confidence in other agents is updated only when they provide opinions or recommendations, a strange phenomenon could happen: agents who never give advice (i.e., because they are not online or because they do not provide new information) never leave the contact list. This phenomenon can result in a contact list full of useless agents. In order to study the effect of this phenomenon in the recommendation process, a decreasing factor χ has been introduced in the profile adaptation process. This parameter gives us the ability to progressively lose confidence in reliable agents in order to ignore them in the future, unless they provide useful information. Thus, for every agent a_{e_i} in the contact list of agent a_q, its trust value $t_{q,i}$ is decreased with the following formula:

$$t_{q,i} = \chi * t_{q,i} \tag{4.5}$$

where the trust decreasing factor χ is a value lower than 1.

Therefore, thanks to this adaptation, reliability leaves the initial sense of similarity and progressively acquires the trust scent of multi-agent systems.

4.6 Evaluation Methods

This section presents the methodology used to evaluate our trust model and filtering methods. Particularly, how results have been acquired and which measures have been used to evaluate them.

4.6.1 Evaluation Metrics

A set of metrics is proposed in order to evaluate recommender systems: precision , recall, f-measure, fallout, friendship and diversity. All these measures are explained in section 3.5.1, except friendship.

Friendship

Friendship represents the percentage average of friends a recommender agent has over time and is calculated as follows:

$$H = \frac{\sum_{i=0}^{k} |F_i|}{NF * k} \tag{4.6}$$

where F_i is the number of friends on day i, NF is the total number of possible friends and k is the number of days.

Friendship is a very important measure since it allows us to understand the degree of collaboration in open environments. Moreover, with this measure we can see the effect of the parameters used in order to generate the contact list.

4.6.2 Results Acquisition

An extended version of the profile discovering evaluation procedure (see section 3.5.2) has been designed in order to simulate a multi-agent system (MAS) of collaborative recommender agents instead of an isolated recommender agent. If the current techniques proposed in the state-of-the-art do not allow the proper evaluation of single recommender agents, neither are they valid for evaluating a MAS of collaborative recommender agents. Thus, we propose the *"profile discovering evaluation procedure with collaboration"*. The main idea of this new technique is essentially the same as profile discovering but it takes into account the opinions and recommendations of other recommender agents in the system as well. The simulation is also performed cycle by cycle. At every cycle new agents enter into the simulation process, the various agents try to make new friends and each agent recommends new items based on the simulated user profile with collaboration from the other agents. Thus, the profile discovering evaluation procedure with collaboration consists of the following

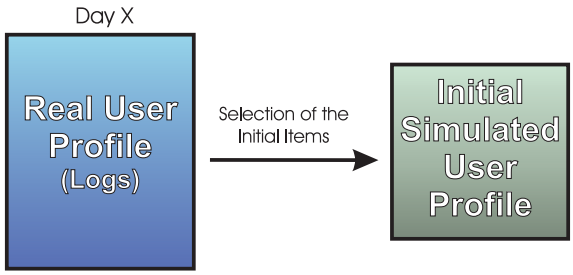


Figure 4.1: Initial Profile Generation.

steps:

1. Initial Profile Generation: as in the profile discovering procedure, an initial simulated user profile is generated as from the real user profile contained in the logs (see Figure 4.1). Each real user profile has a day assigned, which is the cycle the simulated agent enters into the system. Thus, the simulator has to check at every cycle whether there is any new entry and generate the corresponding initial profile.

2. Contact List Generation: each recommender agent has to generate a contact list of reliable friends where reliability expresses similar interests and preferences. Then, only recommender agents will ask for opinions or advice from the most reliable agents in the contact list. Particularly, our agents look for new reliable friends through the playing agents procedure (see section 4.5.2). Thus, the simulator emulates the process where each agent looks for new reliable friends at every cycle (see Figure 4.2). The method to select the group of items to compare and the minimum trust value a reliable friend must have, are parameters of the simulator.

3. Recommendation Process: at every cycle, each agent recommends new items based on the simulated user profile and with the collaboration of the most reliable agents contained in its contact list (see Figure 4.3). The collaboration of reliable agents is achieved through the opinion-based filtering method and through collaborative recommendations based on trust. The opinion-based filtering method consists of the validation of a recommendation asking for an opinion from reliable friends when the given agent is not sure. The recommendation algorithm and the thresholds defining when items are recommended to the user and when the opinion-based filtering method has to be applied

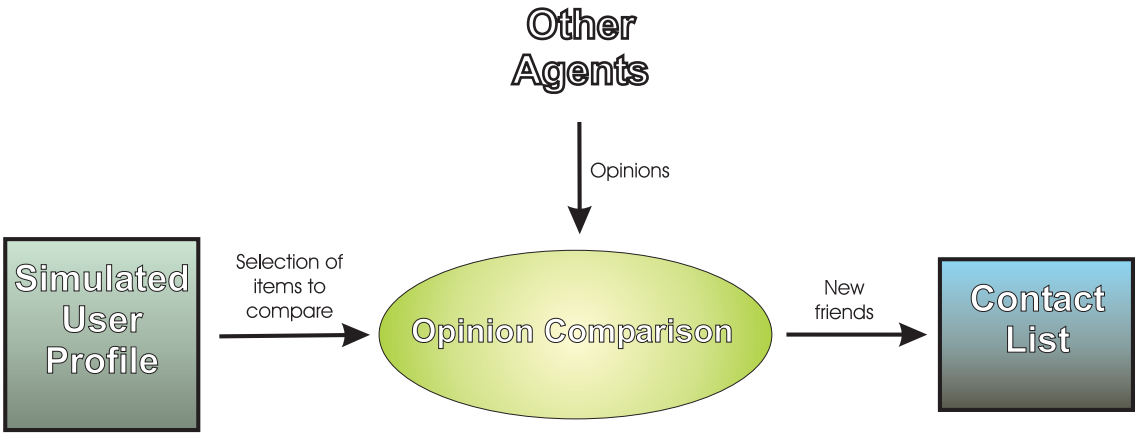


Figure 4.2: "Playing Agents" Procedure.

are parameters of the system. Furthermore, agents can also receive collaborative recommendations by directly asking for interesting items of their reliable friends. Finally, as in the profile discovering procedure, after each recommendation, the simulator checks its success based on the user's assessments contained in the real user profile.

4. Profile Adaptation: besides classifying the recommendation as a success or a failure, the simulator has to adjudge on the collaboration of the other agents. In other words, if some reliable friends have recommended to an agent a certain item that has resulted in a failed recommendation, the given agent has to decrease the trust in those friends. Likewise, if the other agents have recommended an item that results in a successful recommendation, an increment of their trust is necessary. The parameters controlling the modification of trust values of agents in the contact list are parameters of the system.

In summation, the profile discovering procedure with collaboration consists of the entrance of the recommender agents into the MAS, the interaction among them in order to know each other, the recommendation process where each agent recommends new items with the collaboration of their reliable friends, and the adaptation of the user profile in order to improve the performance of the agent in the future. Finally, when the simulation duration is exhausted, several metrics are analysed and the results are presented.

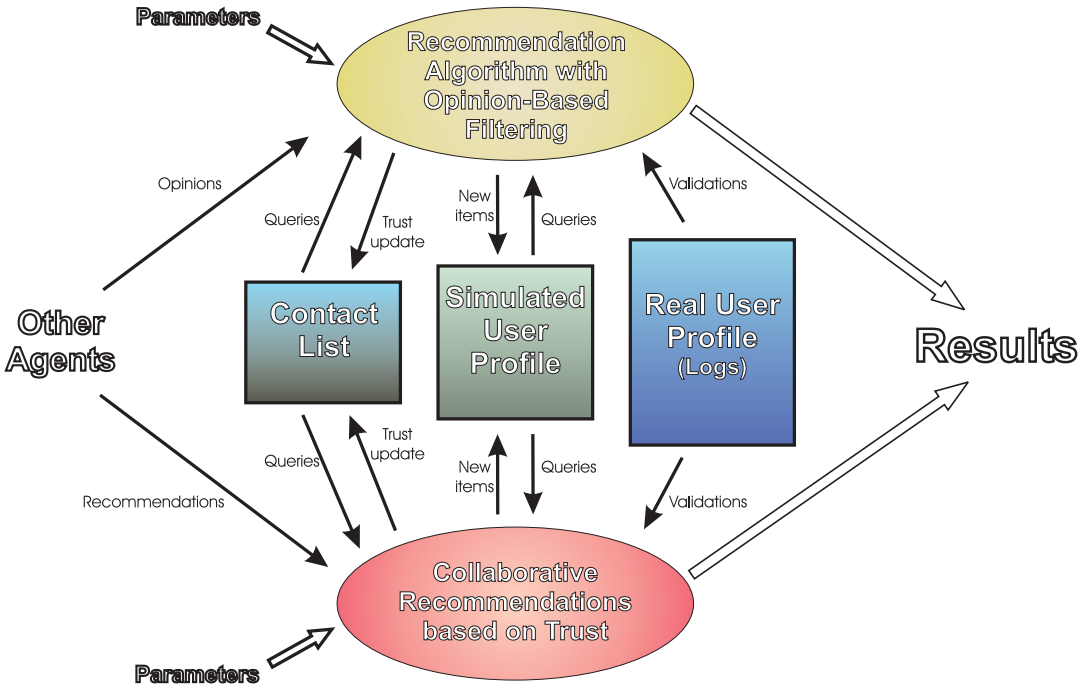


Figure 4.3: Recommendation Process in Profile Discovering with Collaboration.

4.7 Experimental Results

The opinion-based filtering method (OBF) and the collaborative filtering method based on trust (CFT) have been implemented in GenialChef as well (see section 3.6). The system was evaluated using the "profile discovering procedure with collaboration" and all the evaluation measures explained in section 4.6.1 have been computed in relation to different levels of acceptance in contact lists.

The results of the simulations are analysed in this section with the help of several charts. The y-axis refers to the evaluation measures, namely precision, recall, f-measure, fallout, friendship and diversity. All the measures, except diversity, are represented in percentages. The x-axis refers to the different levels of acceptance in contact lists; that is, the trust threshold (see, for example, Figure 4.4). The trust threshold, ω, has two functions:

- ω is the parameter used in the playing agents procedure in order to control how tolerant the agents are when selecting other agents as reliable; that is, the minimum trust agent A must have in another agent, B, so that B belongs

to A's contact list.

- ω is the parameter used in the recommendation process in order to know who are the most reliable friends; that is, the minimum trust agent A must have in another agent, B, so that A takes into account B's opinions and advice.

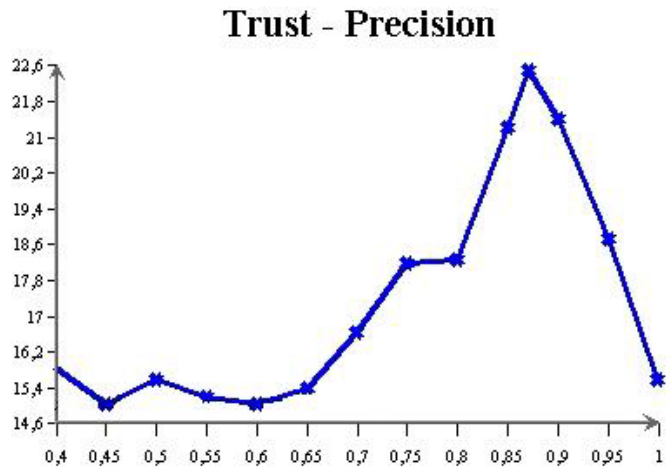
In particular, 13 different levels are analysed (every 0.05 from 0.4 to 1.0). The ω values range from accepting almost every agent ($\omega = 0.4$); that is, agents relying on all the others, to excluding all agents ($\omega = 1.0$); that is, no collaboration is performed. This simple representation allows us to find out the effect of collaboration on each evaluation measure.

In order to avoid anomalous data, as in the forgetting mechanism evaluation, simulations have been performed with 40 different real profiles, most of them obtained after a real testing period of 2 months.

Below, we present the general results obtained with the profile discovering procedure with collaboration. The information filtering methods proposed are then compared to the existing ones. Since OBF and CFT are implemented on the basis of a content-based filtering method (CBF) with the forgetting mechanism, we next analyse the effect of the collaborative world on the forgetting mechanism. Finally, we validate the outstanding performance of the OBF and the CFT by analysing the results obtained when the different parameters of the simulations are changed.

4.7.1 General Results

In order to analyse the performance of the system from a global perspective when the OBF and the CFT methods are applied, several simulations were carried out. Since the methods proposed have been implemented over the CBF method with the forgetting mechanism, we want to point out that the following parameters were used in order to configure the CBF method:



PARAMETERS [a]

$\tau^+ = 0.7$ $\tau^- = 0.3$
$Playing0$
$\chi = 1.0$
$\varphi = 0.85$

[a] the parameters used in the simulation giving the chart on the left as a result.

Figure 4.4: Precision of the system.

PARAMETER	VALUE
Recommendation Algorithm	CBR/E
Initial Item Set	$Training3$
Drift Threshold	$\xi = 0.7$
Drift Decreasing Factor	$\beta = 0.98$
Rewarding/Penalasing Factors	$\lambda = 1.05$ $\sigma = 0.95$
Success/Failure Thresholds	$\phi = 0.5$ $\gamma = 0.499$

These CBF parameters have shown to be the best in order to maximise the performance of the recommender system without collaboration. Besides, another set of simulation parameters regarding the MAS were selected as representative in order to analyse the performance of the system. Specifically, the simulations were performed with the doubt thresholds $\tau^+ = 0.7$ and $\tau^- = 0.3$; with the playing item set $Playing0$; with the trust decreasing factor $\chi = 1.0$ and finally, with the trust modifying factor $\varphi = 0.85$. These parameters are described in detail in section 4.7.4.

Using these simulations, we analysed precision, recall, f-measure, fallout, friendship and diversity measures, the details of which are given below.

Precision

Figure 4.4 shows the precision of the system when different trust thresholds are analysed. We can see that precision exhibits a pronounced rise when agents are

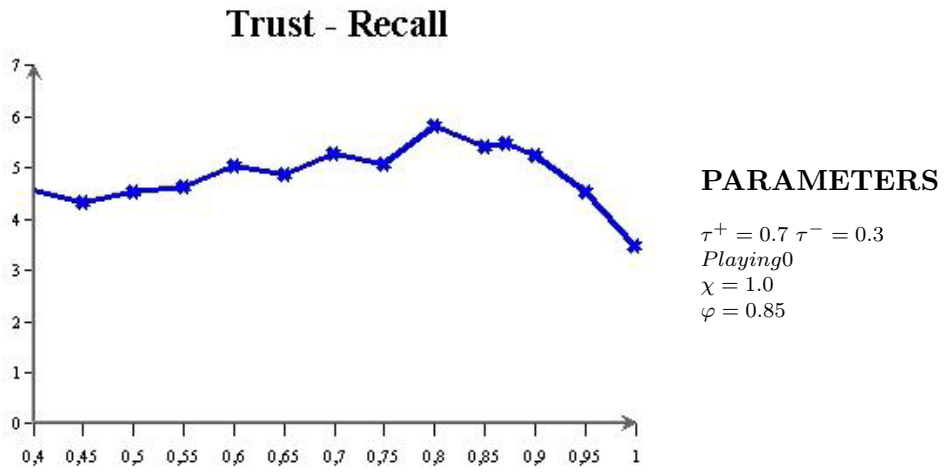


Figure 4.5: Recall of the system.

quite selective (between $\omega = 0.85$ and $\omega = 0.9$); that is, when only agents with a high trust value are accepted as reliable. However, the system presents very bad precision when many agents are contained in the contact list, even worse than when there is no collaboration ($\omega = 1.0$). Therefore, we can say that the information filtering methods proposed exhibits an outstanding precision, provided that, agents are very selective when they choose their reliable friends.

Recall

The system also exhibits better recall when agents are selective (see Figure 4.5), although to a lesser degree than precision. Reasonably, the recall of the system is higher when there is collaboration (any ω except $\omega = 1.0$) than when there is not ($\omega = 1.0$), since reliable agents provide new information to the recommendation process. Thus, the minimum recall is found when $\omega = 1.0$. As long as agents are less selective, recall increases sharply until $\omega = 0.8$, the maximum recall, then recall decreases progressively. A priori, we expected that as agents became less selective, more agents were contained in contact lists and, therefore, recall was higher. However, our experiments show that from $\omega = 0.8$ to $\omega = 0.4$ recall slightly decreases. The reason for this behaviour is that as more agents are contained in contact lists, more diverse opinions and advice are provided. Since confidence values used to decide whether to recommend are obtained by aggregating the information provided by the agents in the contact list, as more agents are enquired, more diverse values

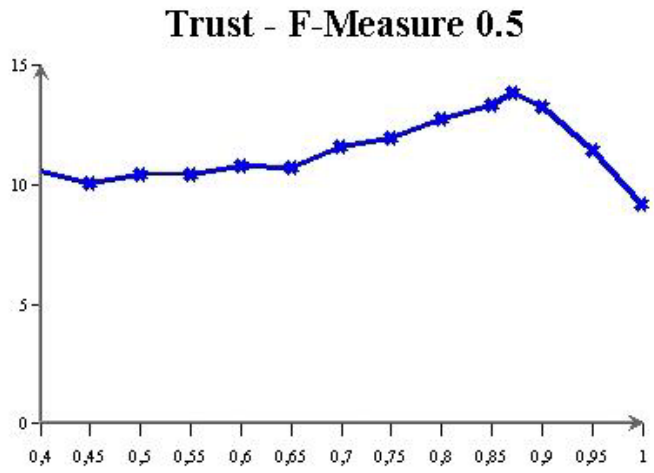


Figure 4.6: F-Measure of the system when b=0.5.

are aggregated and, generally, the result is a mean value that seldom overcomes the doubt threshold τ^+.

F-Measure

Figures 4.6 to 4.8 show the f-measure of the simulation when precision and recall are ponderated. Of course, if precision and recall are higher when agents are selective, the f-measure of the system is also higher. When precision is twice as important as recall (see Figure 4.6), the f-measure is clearly higher, between $\omega = 0.85$ and $\omega = 0.9$. However, when recall is more important than precision (see Figure 4.8), this interval increases and the maximum f-measure is found when $\omega = 0.8$, since this trust threshold has maximum recall. Therefore, we can conclude that regarding precision and recall together, the performance of the system is maximum when the trust threshold is confined between 0.8 and 0.9; that is again, when agents are quite selective.

Fallout

Figure 4.9 shows the fallout of the system. Most of the fallout values are found in the [3-4] interval. We can observe a maximum fallout value when $\omega = 1.0$ and a minimum value when $\omega = 0.5$. Thus, we can say that the collaboration of reliable agents reduces fallout. However, since the difference between values is very low, we

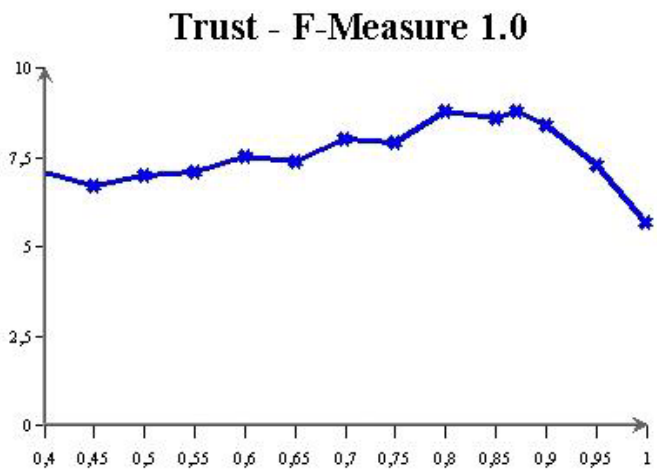


Figure 4.7: F-Measure of the system when b=1.0.

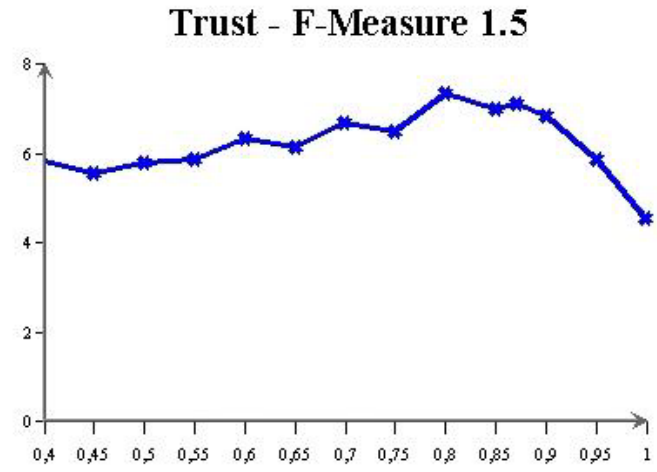


Figure 4.8: F-Measure of the system when b=1.5.

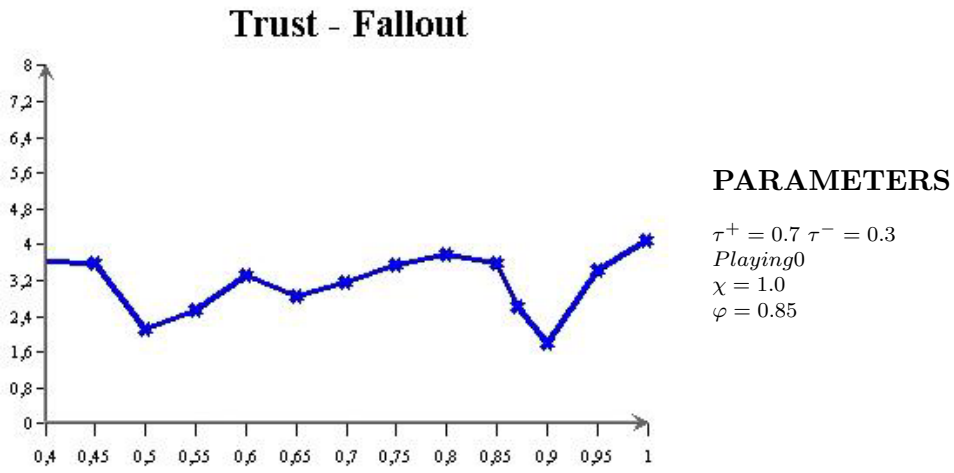


Figure 4.9: Fallout of the system.

believe that the magnitude of tolerance to rely on other agents does not affect the fallout measure.

Friendship

Certainly, the trust threshold strongly affects the percentage of agents in the contact list (see Figure 4.10). The higher the trust threshold, the smaller the number of agents contained in the contact list. For this reason, when $\omega = 0.4$, almost all the available agents are part of the contact lists and, when $\omega = 1.0$, contact lists are empty. When the system achieves the best precision ($\omega = 0.87$), only an average of 11% of agents are contained in the different contact lists, and when the best recall is ($\omega = 0.8$), the average increases up to 26%. These percentages are important in order to know that the best performance of the system is achieved when only a few agents are considered as reliable.

Diversity

The system loses diversity slightly if the number of contact agents decreases (see Figure 4.11). This is perfectly reasonable since with many agents providing recommendations, the information contained in the user profiles will be more diverse. The same happens in the real world; a larger number of people can offer more diverse recommendations than a smaller group. However, this increase of diversity has been

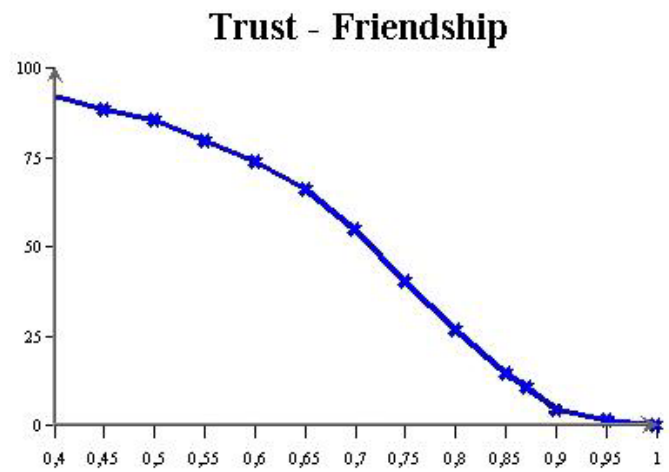


Figure 4.10: Friendship of the system.

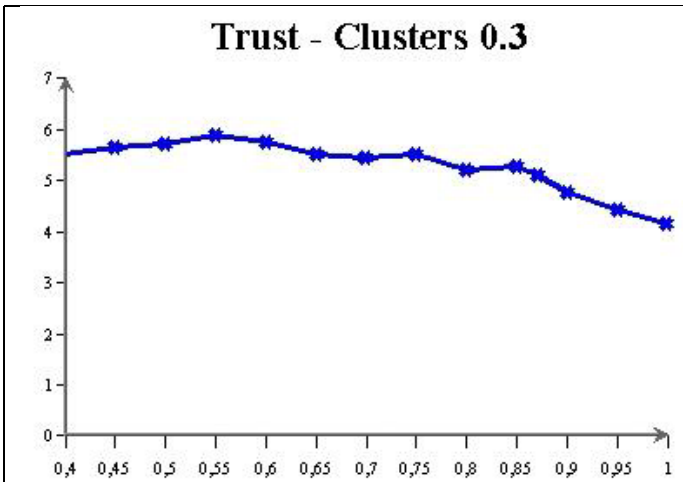


Figure 4.11: Diversity of the system.

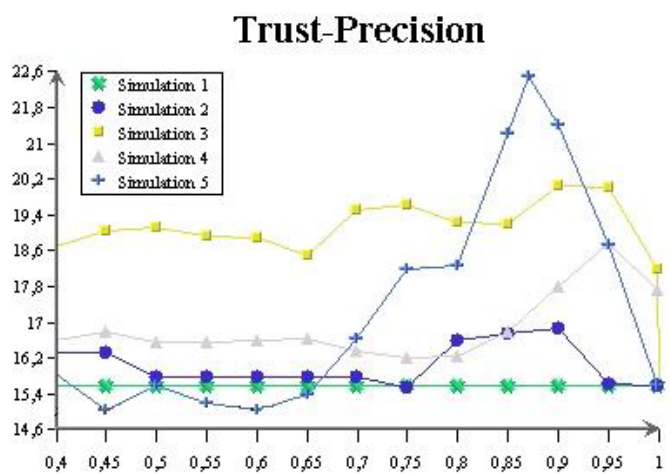


Figure 4.12: Precision of the system with different information filtering methods.

shown to be unproductive, since the best precision and recall are obtained when the diversity is lowest.

4.7.2 Information Filtering Methods

The OBF and the CFT have been proposed as a solution to the shortcomings that arise when recommender systems want to take advantage of the collaborative world. In this section, we will demonstrate that the information filtering methods proposed maintain, and even improve, the performance of the existent methods.

Figures 4.12 to 4.17 show the precision, recall, f-measure, fallout, friendship and diversity measures of the system with different information filtering methods (Simulation1 to Simulation4). In order to make these simulations comparable, we have used the same simulation parameters:

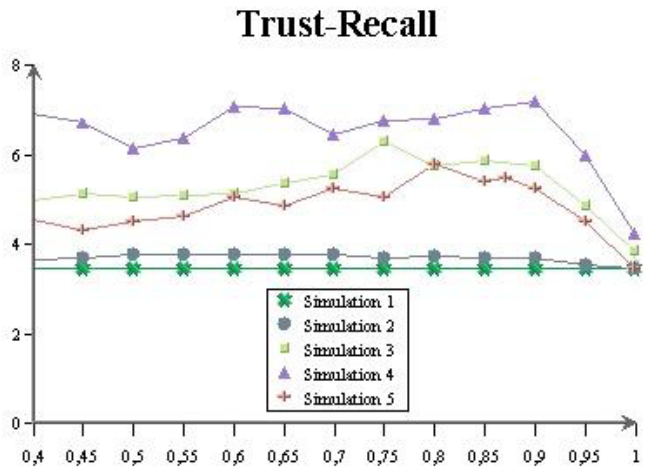


Figure 4.13: Recall of the system with different information filtering methods.

PARAMETER	VALUE
Recommendation Algorithm	CBR/E
Initial Item Set	$Training3$
Drift Threshold	$\xi = 0.7$
Drift Decreasing Factor	$\beta = 0.98$
Rewarding/Penalasing Factors	$\lambda = 1.05$ $\sigma = 0.95$
Success/Failure Thresholds	$\phi = 0.5$ $\gamma = 0.499$
Playing Item Set[1]	$Playing3$
Doubt Thresholds[1]	$\tau^+ = 0.7$ $\tau^- = 0.3$
Trust Decreasing Factor[1]	$\chi = 1.0$
Trust Modifying Factor[1]	$\varphi = 0.85$

[1] These parameters are deeply explained in section 4.7.4

Simulation1 was performed with the content-based filtering method (CBF) proposed in chapter 3, where a forgetting mechanism was used in order to improve the performance of the system. However, a little difference can be found between this CBF and the one implemented in the simulation: the incorporation of the doubt thresholds (see section 4.3.4). Thus, only recommendations with a confidence value $\tau^+ \geq 0.7$ are recommended to the user. All the graphs representing the simulation with the CBF are totally flat, since no collaboration is performed.

Simulation2 integrates the OBF proposed in section 4.3.2 as a complement to the CBF method of Simulation1. When the confidence value of a recommendation

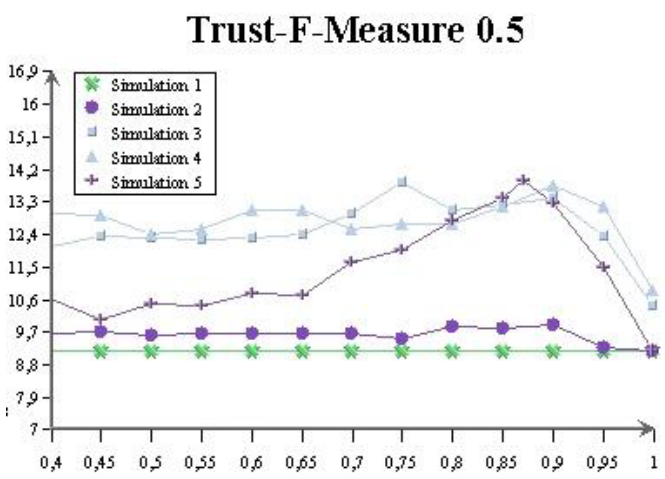


Figure 4.14: F-measure of the system with different information filtering methods when b=0.5.

is confined between doubt thresholds $\tau^+ = 0.7$ and $\tau^- = 0.3$, the opinions of reliable friends are incorporated into the recommendation process. The precision graph presents a slight improvement (see Figure 4.12). The highest precision value can be found between $\omega = 0.8$ and $\omega = 0.9$; that is, when agents are quite selective when adding new friends to the contact list. However, the difference of precision in this range is not very high. This is mainly due to the fact that only when the majority of the best friends' opinions are favourable, the confidence value overcomes τ^+ and the item is recommended to the user. Usually, the best friends' opinions modify the confidence value significantly but, only occasionally does this modification result in a value over τ^+. Therefore, recall is also slightly higher (see Figure 4.13). Since recall and precision are improved, we can conclude that when the best friends' opinions are favourable and the confidence value overcomes τ^+, recommendations are generally successful. With regard to fallout and diversity, both measures are very similar to Simulation1 (see Figures 4.15 and 4.17).

Simulation3 integrates a collaborative filtering method (CF) as a complement to the CBF and the OBF methods of Simulation2. CF is the typical collaborative method proposed in the bibliography, where, before recommending new items, all user profiles are matched in order to find out similar interest. Then, based on these similar profiles, new recommendations are made. Figure 4.12 shows that the precision of the system is drastically improved in Simulation3. As many papers claim [Balabanovic 97, Pazzani 99, Good 99], a hybrid approach between content

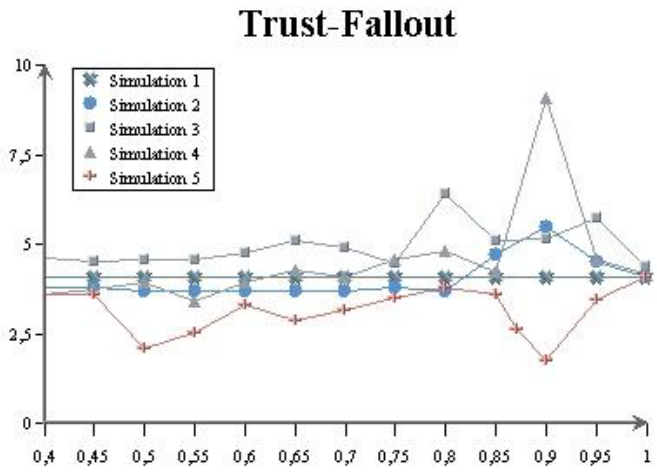
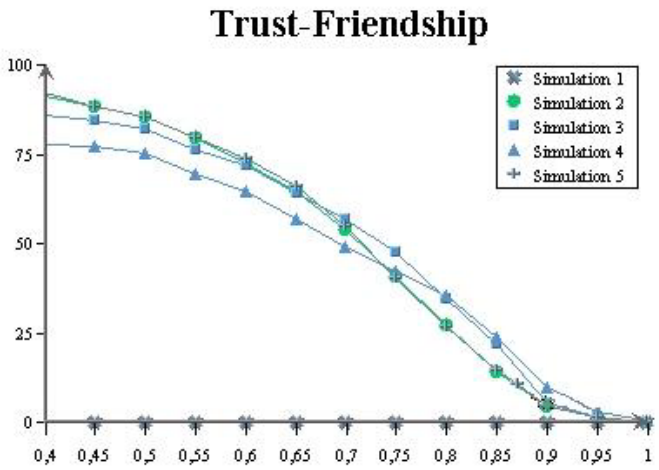


Figure 4.15: Fallout of the system with different information filtering methods.

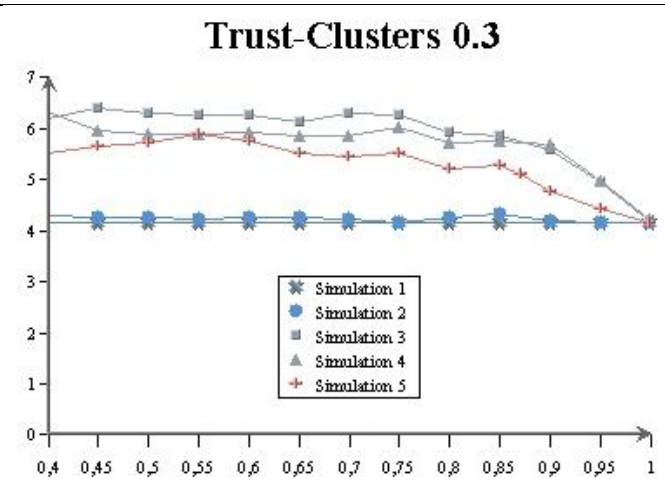
and collaborative filtering methods has an outstanding performance. Furthermore, we want to note that precision remains quite a bit higher even when many agents are considered reliable. Recall is also improved (see Figure 4.13), although fallout also increases a little (see Figure 4.15). At any rate, we can consider that the general performance of the system is sharply improved.

However, the CF method is unfeasible. If thousands of users make use of the recommender system, a profile matching cannot be performed at every cycle. With this in mind, we have designed another simulation scenario. Simulation4 integrates a feasible collaborative filtering method (CF') over the CBF and the OBF methods in Simulation2. CF' works like CF, but the profile matching is carried out every 20 cycles instead of every cycle. We assume that 20 cycles is a reasonable time interval between profile matchings. The results of this simulation show that when many agents are accepted as reliable friends, the precision of the system decreases in such a way that is very similar to the precision in Simulation1 (see Figure 4.12). As agents become more selective when adding reliable friends to the contact list, precision increases, although it never reaches the precision in Simulation3. This is the simulation with the highest recall, although the system preserves a fallout value near the CBF and the diversity is very similar to the CF approach in Simulation3. We can consider that the performance of this simulation is worse than the CBF + OBF + CF, although better than the CBF + OBF.



Simulation1 - CBF
Simulation2 - CBF+OBF
Simulation3 - CBF+OBF+CF
Simulation4 - CBF+OBF+CF'
Simulation5 - CBF+OBF+CFT

Figure 4.16: Friendship of the system with different information filtering methods.



Simulation1 - CBF
Simulation2 - CBF+OBF
Simulation3 - CBF+OBF+CF
Simulation4 - CBF+OBF+CF'
Simulation5 - CBF+OBF+CFT

Figure 4.17: Diversity of the final case bases with different information filtering methods.

Finally, Simulation5 integrates the CFT proposed in section 4.4 as a complement to the CBF and the OBF methods of Simulation2. This is our hybrid approach proposed for collaborative recommender agents. The CFT method consists of adding new friends with an initial trust value through the playing agents procedure and then, depending on the success of their recommendations, updating their trust values according to the trust model proposed. We can observe in Figure 4.12 that Simulation5 is the one with the highest precision between $\omega = 0.85$ and $\omega = 0.9$; that is, when agents need a high trust value to be incorporated into the contact lists. The precision in this interval is even higher than the precision when the optimal collaborative filtering method (CF) is applied. However, when many agents are considered as reliable friends, precision drastically decreases and even reaches values lower than in Simulation1, when no collaboration is performed. This simulation also has high recall (see Figure 4.13), resulting in a high f-measure in the given interval (see Figure 4.14). Moreover, the fallout from the system is lower than in Simulation1 (see Figure 4.15). Finally, the diversity of the final case bases is very similar to Simulation3 and Simulation4 (see Figure 4.17).

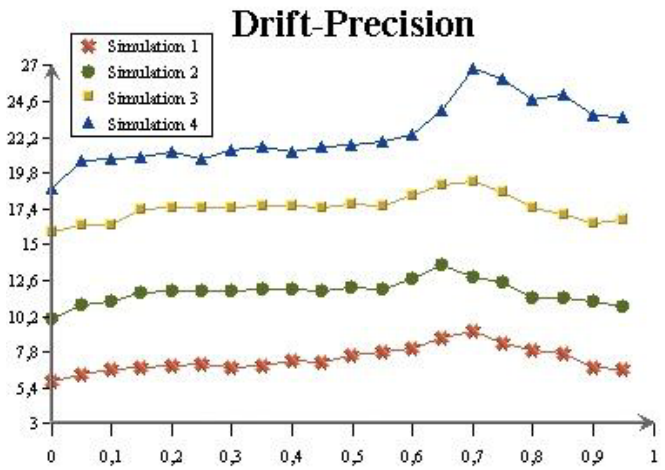
Therefore, we can conclude that the proposed information filtering method based on trust performs better than existing ones, provided that only recommender agents with high trust values are kept in their contact lists.

4.7.3 Forgetting Mechanism

The OBF and the CFT have been proposed as a complement to the content-based filtering method (CBF) which improves the performance of a recommender system. Since our proposal of CBF incorporates a forgetting mechanism, this section will demonstrate that the inclusion of the collaborative world in recommender systems does not affect the outstanding performance of the forgetting mechanism.

In order to do that, four simulations were performed. Figures 4.18 to 4.24 show the results of these simulations from the point of view of different measures. Note that, in these figures, the x-axis represents the different levels of forgetfulness; that is, how much is forgotten by the forgetting mechanism.

Simulation1 gave us the results of the system when agents are simulated individually and when the forgetting mechanism based on the drift attribute is applied over the CBF method. Cases in case-based profiles are weighted taking into account



Simulation1 - *training*0, isolated agent, $\tau^+ = 0.5$, $\tau^- = 0.5$

Simulation2 - *training*0, isolated agent, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation3 - *training*0, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

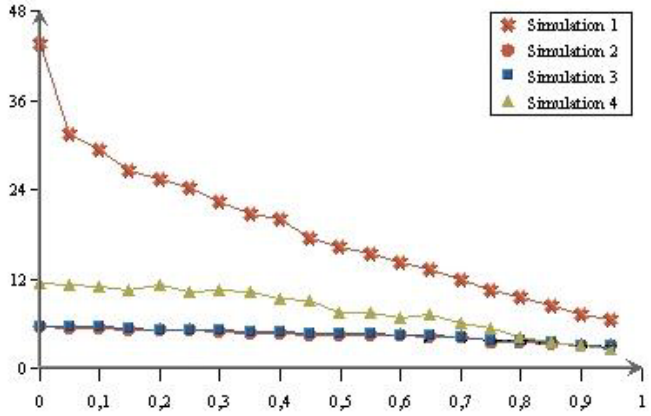
Simulation4 - *training*3, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Figure 4.18: Precision of a MAS of recommender agents with the forgetting mechanism.

their relevance in the recommendation process. Then, irrelevant cases are forgotten to better adapt the profile to the user interests and to solve the utility problem. The results of this simulation are presented in the general results of the forgetting mechanism (see section 3.6.1) and are repeated here for convenience. We want to stress that the precision graph shows the expected bell shape with the maximum value when $\xi = 0.7$.

No doubt thresholds (see section 4.7.4) are considered in the isolated CBF approach; that is $\tau^+ = \tau^- = 0.5$, while Simulation4 utilised $\tau^+ = 0.7$ and $\tau^- = 0.3$. Moreover, Simulation1 generated the initial profiles with the $Training0$ initial item set, while Simulation4 did it with $Training3$. In order to make this result comparable, two more experiments were carried out. The first one in order to see the effect of the doubt thresholds over the CBF method, and the second in order to analyse how a more suitable initial item set affects the performance of the system.

Simulation2 is the same experiment as Simulation1 but with the doubt thresholds $\tau^+ = 0.7$ and $\tau^- = 0.3$. In this simulation, agents were simulated individually with the forgetting mechanism and only items with an interest value over 0.7 were recommended to the user. The results of this simulation show that the precision of the system is improved (40% when $\xi = 0.7$), although recall is considerably lower (-180% when $\xi = 0.7$). Thus, the f-measure, when precision is twice as important as recall, is very similar but worse when recall has more weight. A lower recall



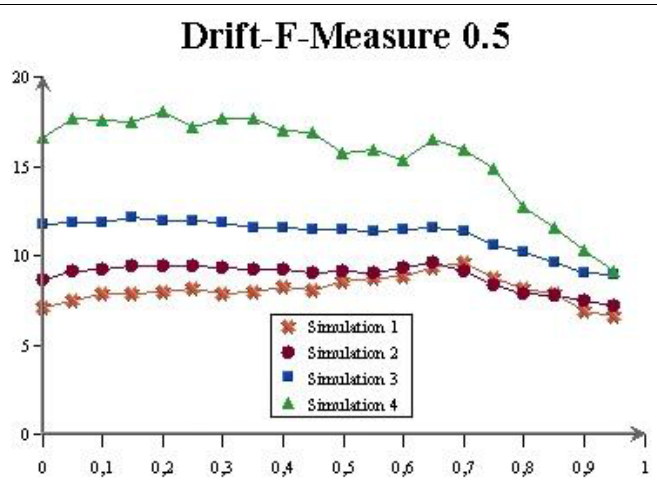
Simulation1 - *training*0, isolated agent, $\tau^+ = 0.5$, $\tau^- = 0.5$

Simulation2 - *training*0, isolated agent, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation3 - *training*0, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation4 - *training*3, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Figure 4.19: Recall of a MAS of recommender agents with the forgetting mechanism.



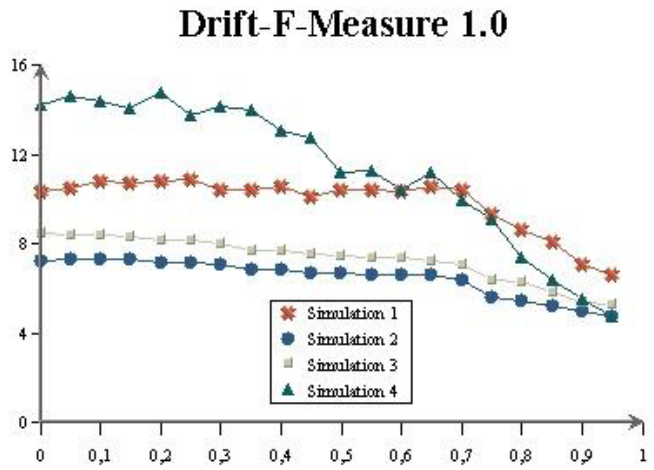
Simulation1 - *training*0, isolated agent, $\tau^+ = 0.5$, $\tau^- = 0.5$

Simulation2 - *training*0, isolated agent, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation3 - *training*0, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation4 - *training*3, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Figure 4.20: F-Measure of a MAS of recommender agents with the forgetting mechanism when b=0.5.



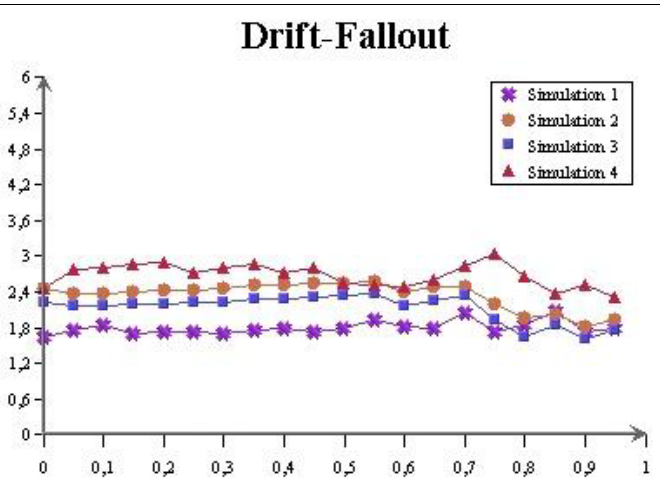
Simulation1 - *training*0, isolated agent, $\tau^+ = 0.5$, $\tau^- = 0.5$

Simulation2 - *training*0, isolated agent, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation3 - *training*0, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation4 - *training*3, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Figure 4.21: F-Measure of a MAS of recommender agents with the forgetting mechanism when b=1.0.



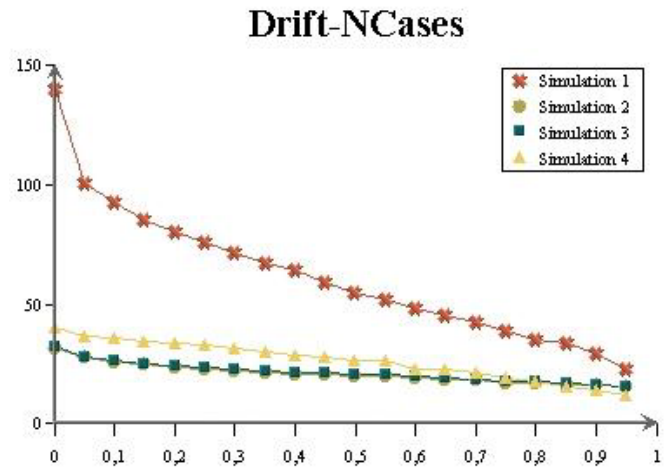
Simulation1 - *training*0, isolated agent, $\tau^+ = 0.5$, $\tau^- = 0.5$

Simulation2 - *training*0, isolated agent, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation3 - *training*0, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation4 - *training*3, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Figure 4.22: Fallout of a MAS of recommender agents with the forgetting mechanism.



Simulation1 - *training*0, isolated agent, $\tau^+ = 0.5$, $\tau^- = 0.5$

Simulation2 - *training*0, isolated agent, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation3 - *training*0, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation4 - *training*3, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

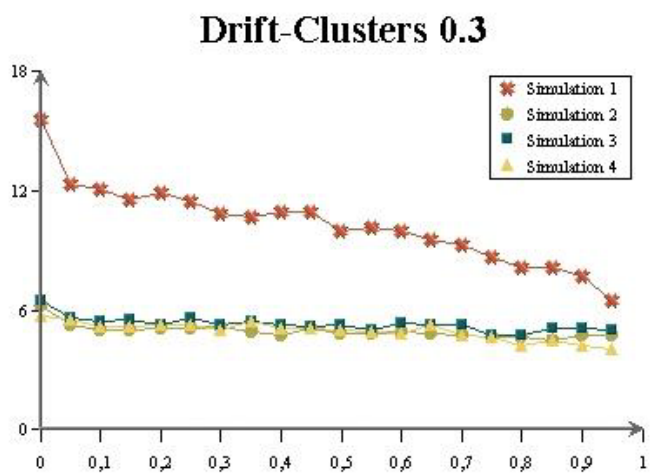
Figure 4.23: NCases of a MAS of recommender agents with the forgetting mechanism.

also produces an important decrease in the average of cases in the case bases over time. The diversity of the final case bases is lower, although Simulation2 loses less diversity than Simulation1.

Simulation3 incorporates the collaborative world over Simulation2; that is, the OBF and the CFT methods have been applied over the CBF method with the forgetting mechanism when $\tau^+ = 0.7$ and $\tau^- = 0.3$. Certainly, the collaborative world provokes better results on recommender agents. In particular, precision increases 50% and recall 2% compared to Simulation2 when $\xi = 0.7$. This produces a better f-measure whatever the weight of precision and recall. Moreover, ncases and diversity are maintained while fallout is slightly better.

The last simulation scenario, Simulation4, differs from Simulation 3 in that the former utilises the *Training*3 initial item set that has proved more useful. This difference provokes an increase of precision around 65% and a recall 40% higher when $\xi = 0.7$. Therefore, f-measure is also improved. Moreover, ncases is slightly high while diversity is maintained. The only worse result is fallout, although the difference is negligible.

Summing up, Simulation1 gives the general results of the forgetting mechanism over an isolated agent. The doubt thresholds in Simulation2 provide recommender agents with more precision though less recall. The incorporation of the collaborative world over Simulation2 by means of the OBF and CFT methods increases precision



Simulation1 - *training*0, isolated agent, $\tau^+ = 0.5$, $\tau^- = 0.5$

Simulation2 - *training*0, isolated agent, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation3 - *training*0, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Simulation4 - *training*3, MAS, $\tau^+ = 0.7$, $\tau^- = 0.3$

Figure 4.24: Diversity of a MAS of recommender agents with the forgetting mechanism.

significantly while preserving recall. And finally, the most suitable initial item set, *Training*3, over Simulation3 provides the best results of the MAS of recommender agents. In all the simulations, the behaviour of the forgetting mechanism, through the different levels of forgetfulness, is the same. In particular, precision outlines a bell-shaped graph with the maximum value when $\xi = 0.7$ and the minimum when the system does not forget at all. Recall decreases as long as the drift threshold increases and fallout remains stable. The number of cases in case bases is reduced and diversity, though lower, is maintained regardless of ξ.

4.7.4 System Parameter Evaluation

In order to validate these general results, thousands of simulations with different user profiles, different trust parameters and different playing item sets were performed. The parameters used in the simulations are summarised in Table 4.4 and the results of the simulations are shown in the following sections.

Doubt Thresholds

The doubt thresholds τ^+ and τ^- define when a recommender agent is sure about a recommendation. When the interest value of a new item is above τ^+, the item is recommended to the user (see section 4.3.4). Moreover, these parameters are also

PARAMETERS	VALUES
User Profiles	$Profile_1$, $Profile_2$, ..., $Profile_{40}$
Trust Thresholds	$\omega = 0.4$, $\omega = 0.45$, $\omega = 0.5$, ..., $\omega = 1.0$
Doubt Thresholds	$\tau^+ = 0.5$ $\tau^- = 0.5$, $\tau^+ = 0.6$ $\tau^- = 0.4$, $\tau^+ = 0.7$ $\tau^- = 0.3$, $\tau^+ = 0.8$ $\tau^- = 0.2$
Playing Item Sets	$Playing0$, $Playing1$, $Playing2$, $Playing3$
Trust Decreasing Factor	$\chi = 1.0$, $\chi = 0.999$, $\chi = 0.99$
Trust Modifying Factor	$\varphi = 0.0$, $\varphi = 0.4$, $\varphi = 0.85$, $\varphi = 1.0$

Table 4.4: MAS Simulation Parameters.

used in order to decide when the agents collaborate by means of the opinion-based filtering method. When the interest value of a new item is confined in $[\tau^+, \tau^-]$, the best friends' opinions are incorporated into the decision process. In particular, four (τ^+, τ^-) pairs were tested:

- $\tau^+ = 0.5$ and $\tau^- = 0.5$: if an item has an interest value above 0.5, it is recommended to the user. Otherwise it is ignored. The opinion-based filtering method is never used.
- $\tau^+ = 0.6$ and $\tau^- = 0.4$: the top doubt threshold is low, although a band of uncertainty has been generated. Items with an interest value confined in [0.4-0.6] have to be verified with reliable friends.
- $\tau^+ = 0.7$ and $\tau^- = 0.3$: items need a rather high interest value to be recommended to the user. An important band of uncertainty means that many recommendations have to be verified with the best friends' opinions.
- $\tau^+ = 0.8$ and $\tau^- = 0.2$: agents only recommend items when they are very sure about it. Most of the time, the opinion-based filtering method is incorporated into the decision process.

A priori, the simulations proposed lead us to believe that the higher τ^+ and the lower τ^- are, the more precision, although with a lower recall. Figures 4.25 to 4.30 show the results obtained when different doubt thresholds are analysed.

Precision. Figure 4.38 shows the precision of the simulations proposed. As expected, the simulation with the highest precision is Simulation1; that is, when $\tau^+ = 0.8$ and $\tau^- = 0.2$. Since agents only recommend new items when they are really sure about it, recommendations often result in successes. On the other hand, Simulation3 and Simulation4 are the ones with the lowest precision, since only a

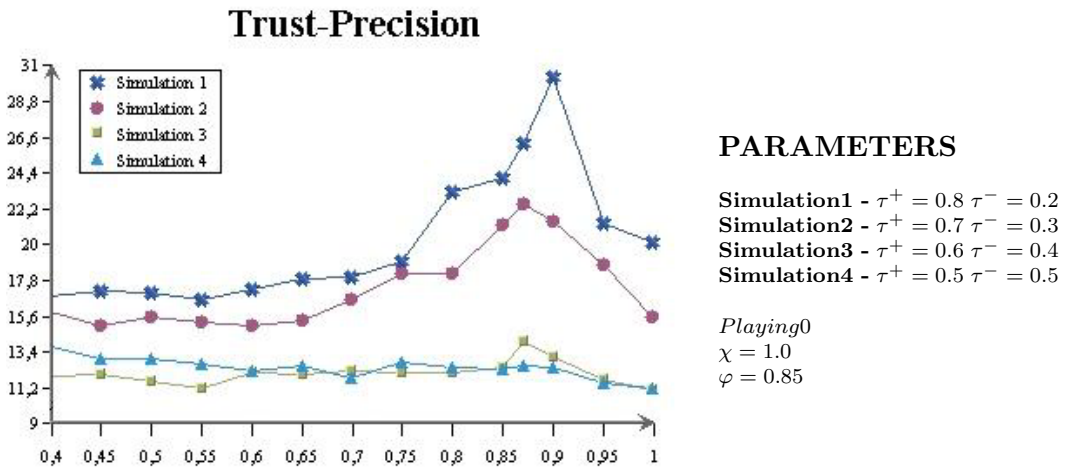
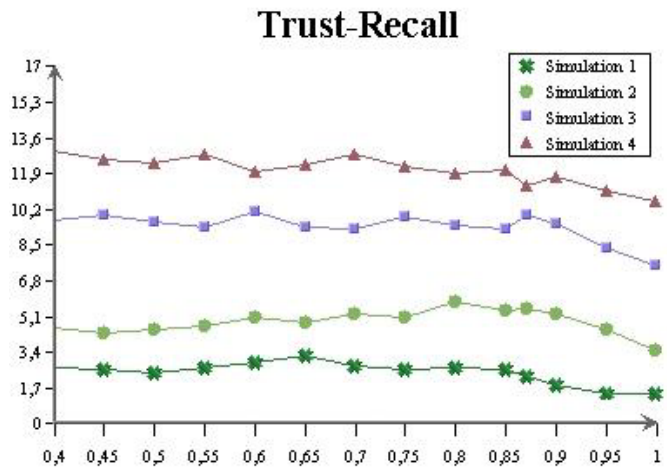


Figure 4.25: Precision of the system with different doubt thresholds.

minimum level of confidence has to be achieved in order to recommend. Therefore, it is important to note that all the simulations, except Simulation4 (no OBF method), present their best precision within the interval $[\omega = 0.85, \omega = 0.9]$. Thus, we corroborate the general result that the best performance is obtained in this interval.

Recall. Reasonably, the doubt thresholds strongly affect the recall of the system. The higher τ^+, the lower the recall. With a high τ^+, only occasionally does the agent find a new item with an interest value above this threshold and, therefore, only a few recommendations are performed. By contrast, when just a minimum level of confidence has to be achieved (when $\tau^+ = 0.5$), most of the new items are recommended to the user.

F-Measure. As expected, the simulations with the highest τ^+ are the ones with the highest precision and the lowest recall. If we analyse precision and recall together, we can see how precision is balanced by recall. For example when b=0.5; that is, when precision is twice as important as recall, Simulation1, which was the simulation with the best precision, becomes the one with the worst f-measure (see Figure 4.27). In this case, Simulation2 is the one with the best f-measure between $\omega = 0.85$ and $\omega = 0.9$. When precision and recall have the same importance (see Figure 4.28), the higher recall in Simulation4 results in the best f-measure.

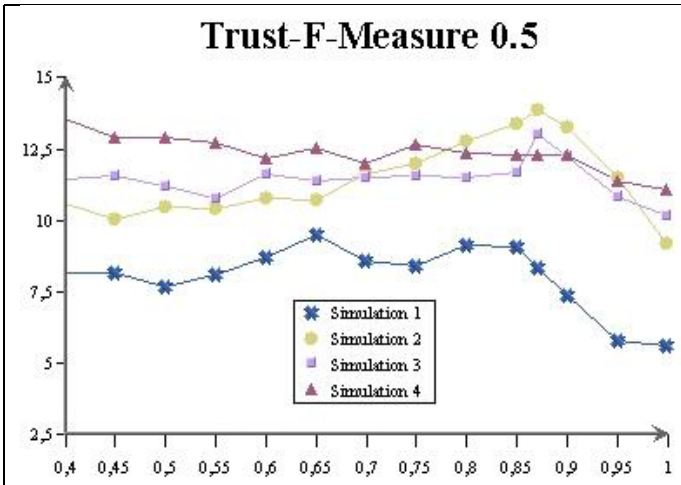


PARAMETERS

Simulation1 - $\tau^+ = 0.8$ $\tau^- = 0.2$
Simulation2 - $\tau^+ = 0.7$ $\tau^- = 0.3$
Simulation3 - $\tau^+ = 0.6$ $\tau^- = 0.4$
Simulation4 - $\tau^+ = 0.5$ $\tau^- = 0.5$

*Playing*0
$\chi = 1.0$
$\varphi = 0.85$

Figure 4.26: Recall of the system with different doubt thresholds.

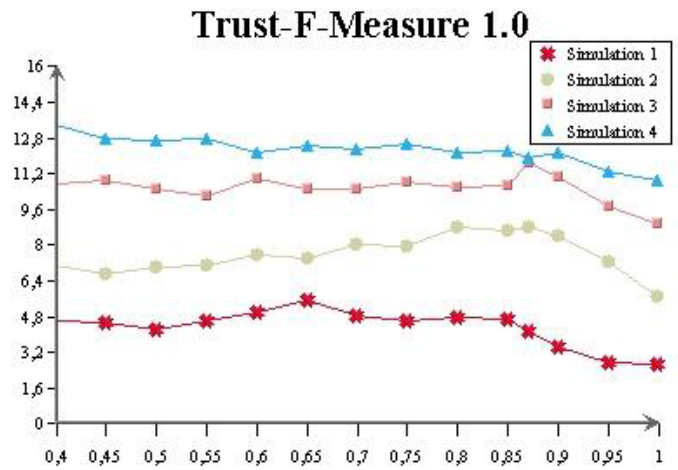


PARAMETERS

Simulation1 - $\tau^+ = 0.8$ $\tau^- = 0.2$
Simulation2 - $\tau^+ = 0.7$ $\tau^- = 0.3$
Simulation3 - $\tau^+ = 0.6$ $\tau^- = 0.4$
Simulation4 - $\tau^+ = 0.5$ $\tau^- = 0.5$

*Playing*0
$\chi = 1.0$
$\varphi = 0.85$

Figure 4.27: F-Measure of the system with different doubt thresholds when b=0.5.

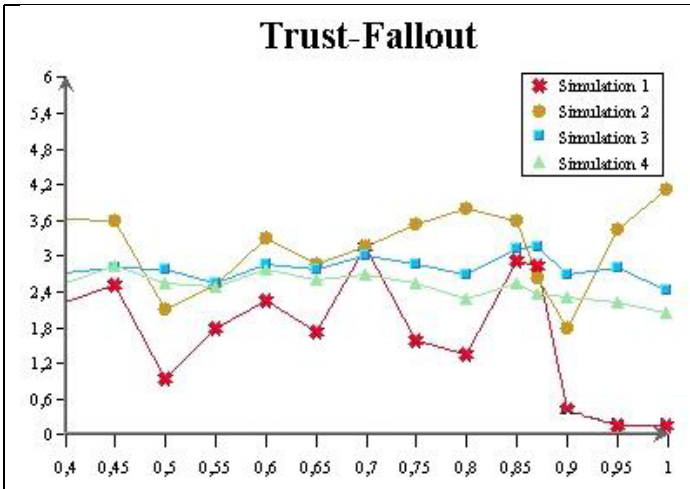


PARAMETERS

Simulation1 - $\tau^+ = 0.8$ $\tau^- = 0.2$
Simulation2 - $\tau^+ = 0.7$ $\tau^- = 0.3$
Simulation3 - $\tau^+ = 0.6$ $\tau^- = 0.4$
Simulation4 - $\tau^+ = 0.5$ $\tau^- = 0.5$

*Playing*0
$\chi = 1.0$
$\varphi = 0.85$

Figure 4.28: F-Measure of the system with different doubt thresholds when b=1.0.



PARAMETERS

Simulation1 - $\tau^+ = 0.8$ $\tau^- = 0.2$
Simulation2 - $\tau^+ = 0.7$ $\tau^- = 0.3$
Simulation3 - $\tau^+ = 0.6$ $\tau^- = 0.4$
Simulation4 - $\tau^+ = 0.5$ $\tau^- = 0.5$

*Playing*0
$\chi = 1.0$
$\varphi = 0.85$

Figure 4.29: Fallout of the system with different doubt thresholds.

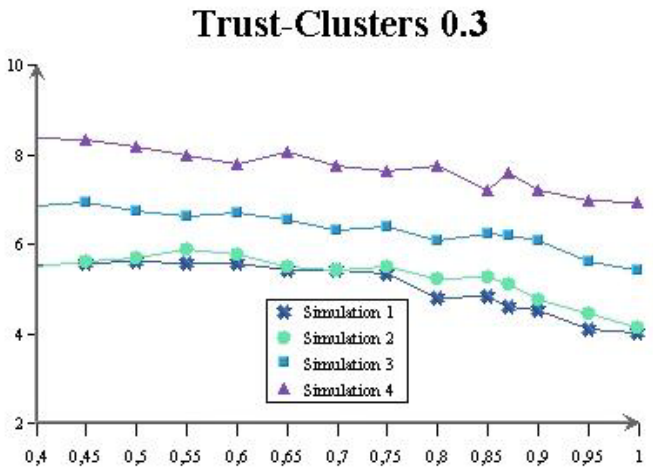


Figure 4.30: Diversity of the system with different doubt thresholds.

Fallout. All the simulations have similar fallout (see Figure 4.29). However, we can consider that Simulation1 has the best fallout, although we have to take into account that it is also the one with the lowest recall.

Friendship. The doubt thresholds do not affect the average percentage of agents contained in contact lists over time, since these parameters are not used in agent selection.

Diversity. Figure 4.44 shows the diversity of the simulations proposed. Due to the correlation between recall and diversity, the simulations with the highest recall are also the ones with the highest diversity in the final user profiles. This is mainly due to the fact that the simulations that recommend fewer items have user profiles with fewer items and, therefore, diversity is affected.

Summing up, the results of the simulations concerning doubt thresholds showed that if we want precise recommender agents, a high τ^+ and a low τ^- has to be selected. Contrarily, if we are interested in recall, a low τ^+ and a high τ^- has to be selected. We believe that a good trade-off is $\tau^+ = 0.7$ and $\tau^- = 0.3$ and, therefore, the ones used in our simulations.

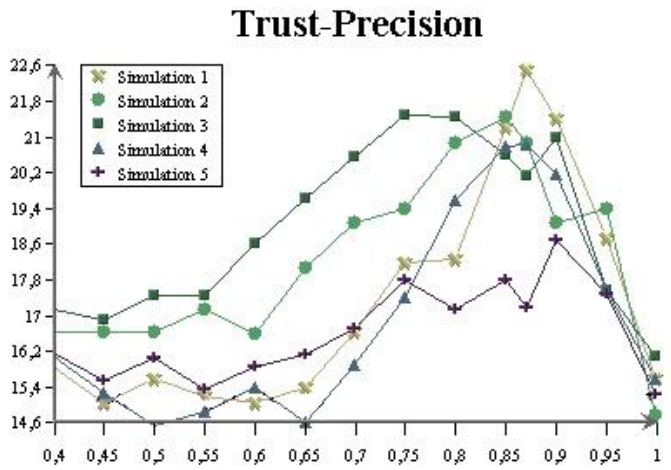
Playing Item Sets

The set of items selected to be compared in the playing agents procedure is a parameter of the simulator. The aim of this section is to analyse the effect of the playing item sets on the performance of the system. Five different sets were simulated:

- *Playing*0: the three items with the highest interest value and the three with the lowest are selected.
- *Playing*1: utilises the item with the highest interest value and the one with the lowest.
- *Playing*2: only the three items with the highest interest value are selected.
- *Playing*3: all the items contained in the user profile are compared.
- *Playing*4: four items in the user profile are selected randomly.

We expect the playing item set to condition the tolerance of the agents when making new friends. For example, we believe that *Playing*1 is more selective than the others since only two items are compared. If agents do not coincide in these items, they cannot be friends. However, *Playing*3 compares all the items and, therefore, it is easier to find out coincidences. The *Playing*4 set was simulated in order to see if selecting which items to compare is relevant to the final results. Figures 4.31 to 4.37 show the results obtained when the different playing item sets are analysed.

Precision. Different precisions are obtained when the different playing item sets are used in order to make new friends (see Figure 4.31). All the simulations show a low precision when no collaboration is performed ($\omega = 1.0$) and when many agents are contained in the contact lists (from $\omega = 0.4$ to $\omega = 0.65$). The best precision is obtained when agents are selective; that is, when the trust threshold is around 0.85. However, depending on the playing item set, the precision is higher. For example, the simulation with the highest precision is when *Playing*0 is utilised. If we compare only the three best items (*Playing*2), the precision of the system is higher when many agents are contained in the contact lists. Finally, *Playing*4 is the playing item set with the lowest precision. This fact makes us realise how important the criteria to select a suitable set of items to compare is.



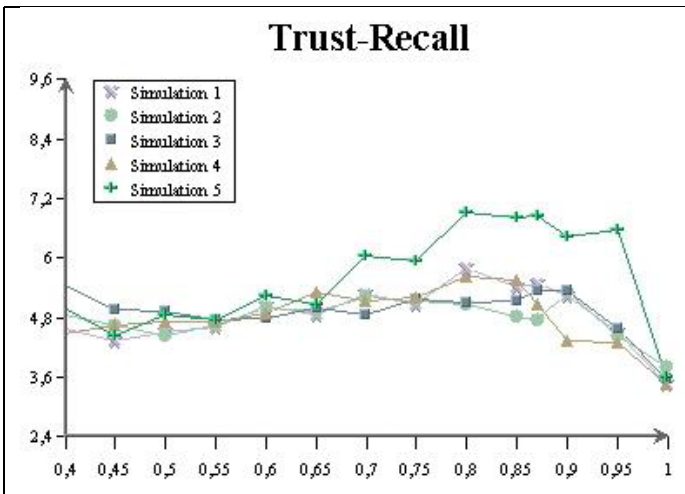
PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$

Simulation1 - *Playing*0
Simulation2 - *Playing*1
Simulation3 - *Playing*2
Simulation4 - *Playing*3
Simulation5 - *Playing*4

$\chi = 1.0$
$\varphi = 0.85$

Figure 4.31: Precision of the system with different playing item sets.



PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$

Simulation1 - *Playing*0
Simulation2 - *Playing*1
Simulation3 - *Playing*2
Simulation4 - *Playing*3
Simulation5 - *Playing*4

$\chi = 1.0$
$\varphi = 0.85$

Figure 4.32: Recall of the system with different playing item sets.

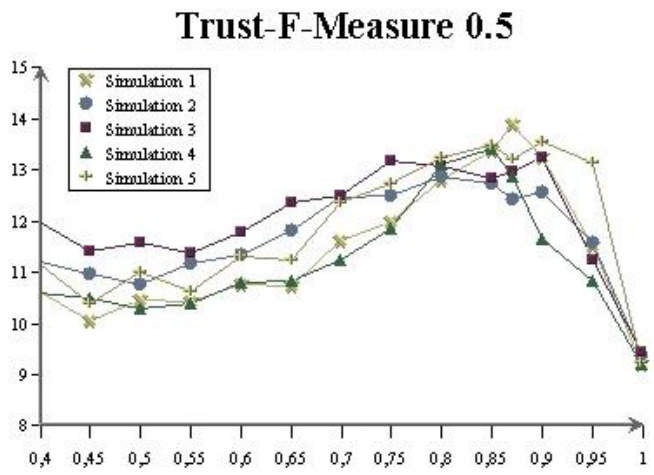
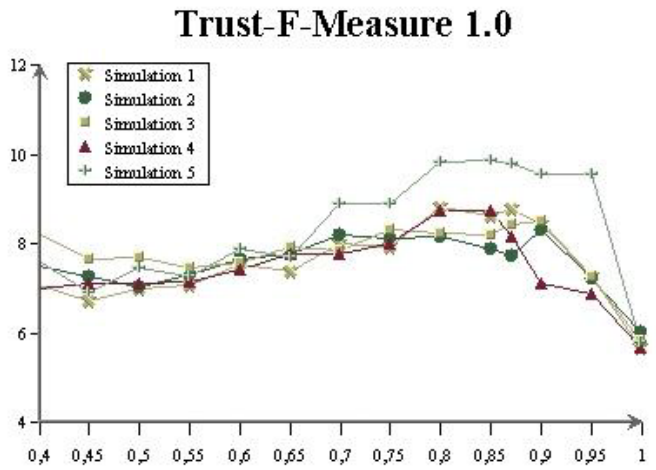


Figure 4.33: F-Measure of the system with different playing item sets when b=0.5.

Recall. The recall of the simulations with different playing item sets is very similar, except for *Playing*4 (see Figure 4.32). They show a slightly better recall when agents are quite selective and the worst recall is found when no collaboration is performed. *Playing*4 has a considerably higher recall between $\omega = 0.65$ and $\omega = 0.95$. This is mainly due to the fact that this playing item set incorporates a high number of agents in the contact lists (see Figure 4.36) and, therefore, agents collaborate to a higher degree.

F-Measure. Figures 4.33 and 4.34 show the performance of the system when precision and recall are analysed together. All the simulations have a similar f-measure when precision is twice as important as recall (see Figure 4.33). However, the highest recall, in the simulation with *Playing*4, results in a higher f-measure when precision and recall are equally weighted (see Figure 4.34).

Fallout. No important differences were found when analysing the fallout of the different simulations (see Figure 4.35). We only want to stress that *Playing*0 performs slightly better, since it has a lower fallout than the other simulations in most of the trust thresholds, especially between $\omega = 0.85$ and $\omega = 0.9$.



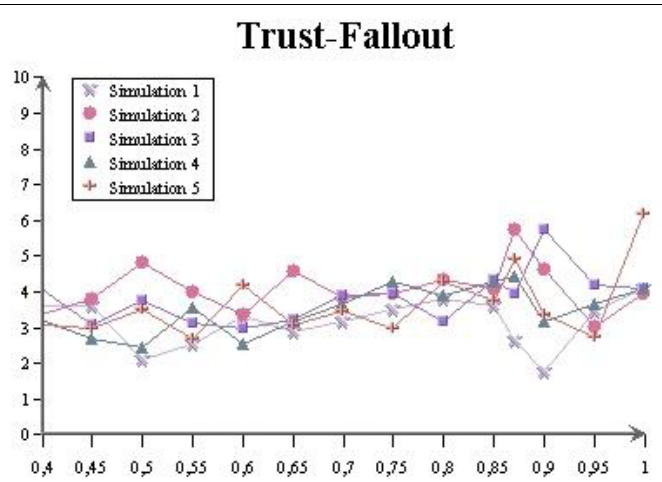
PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$

Simulation1 - *Playing*0
Simulation2 - *Playing*1
Simulation3 - *Playing*2
Simulation4 - *Playing*3
Simulation5 - *Playing*4

$\chi = 1.0$
$\varphi = 0.85$

Figure 4.34: F-Measure of the system with different playing item sets when b=1.0.



PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$

Simulation1 - *Playing*0
Simulation2 - *Playing*1
Simulation3 - *Playing*2
Simulation4 - *Playing*3
Simulation5 - *Playing*4

$\chi = 1.0$
$\varphi = 0.85$

Figure 4.35: Fallout of the system with different playing item sets.

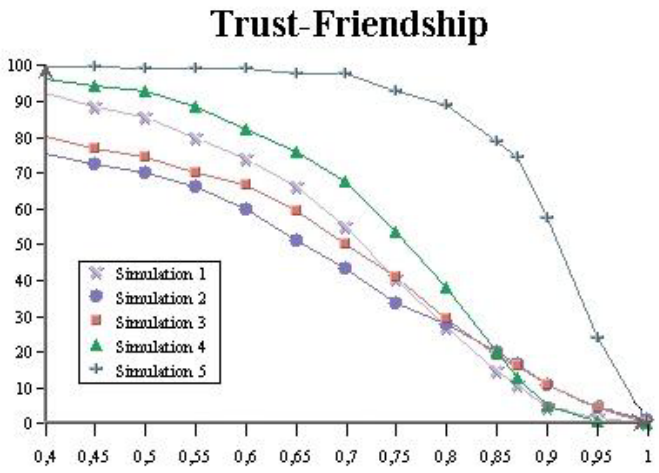


Figure 4.36: Friendship of the system with different playing item sets.

Friendship. The most important difference in the results of the simulations performed with different playing item sets is the average percentage of friends contained in the contact lists over time. Reasonably, the items used in order to compare opinions influence how many friends are considered as similar. Figure 4.36 shows the friendship of the different simulations proposed. The playing item set that most differs from the others is definitively *Playing*4. We consider this method the least selective when adding new friends to contact lists. This is because user profiles contain many items with an indeterminate evaluation; that is, a user interest value around 0.5. Since *Playing*4 selects items at random, there is a high probability that these items will be selected and the users will have similar indeterminate opinions about them. The comparison results with an initial trust value above 0.7, thus, when $\omega = 0.7$ shows almost 100% of the agents contained in the contact lists. The second least selective playing item set is when all the items in the user profile are compared (*Playing*3). In this case, the items with an indeterminate interest are also used in the playing agents procedure, thus, they also have similar profiles. However, we believe that the items with an indeterminate interest value do not define the general interests of the user. For this reason, the other playing item sets contain only the items with the highest and lowest interest value. For example, *Playing*1 contain only the item most preferred and the item the least preferred by the user. This is the simulation with least collaboration, since fewer agents are contained in the contact lists. Therefore, we can conclude that *Playing*1 is the playing item set

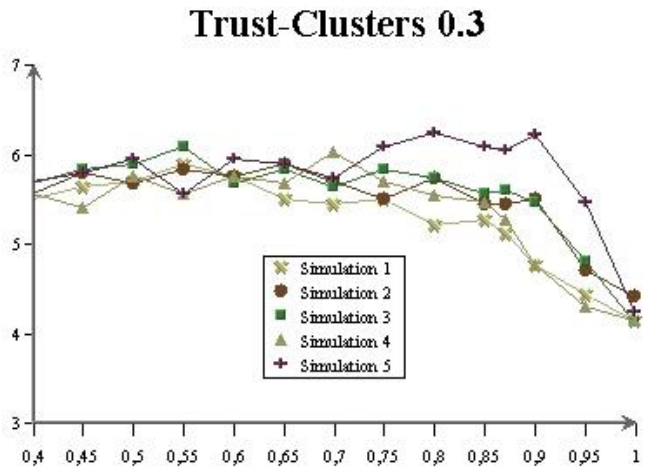


Figure 4.37: Diversity of the system with different playing item sets.

most selective and *Playing*3 and *Playing*4 the ones that most collaborate.

Diversity. All the playing item sets except *Playing*4 produce a similar number of clusters in the user profiles at the end of the simulations. We can see in Figure 4.37 that Simulation5 has a higher diversity between $\omega = 0.75$ and $\omega = 0.95$. As seen before, this simulation is the one with the highest recall and, therefore, produces a higher diversity in the final user profile.

Summing up, the results of the simulations performed with different playing item sets show us the importance of the items used to make new friends. A small set of items is always more selective than a larger group since it is more difficult to coincide with only a few opinions and fewer agents are added to contact lists. A suitable criteria to select which items to compare is also very important. Selecting the items most and least preferred by the user has proven a useful method.

Trust Decreasing Factors

At every cycle, the trust decreasing factor χ decreases the trust of the agents contained in the contact list. The aim of this parameter is to progressively lose reliability on agents who do not provide information (see section 4.5.3). Agents providing information are judged according to the success of their recommendations. But, if

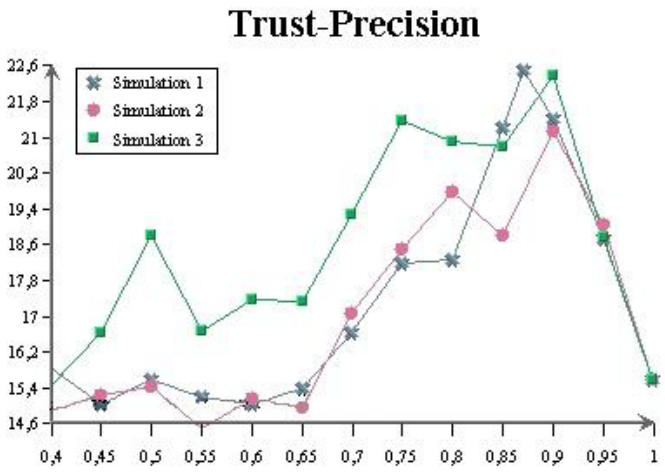


Figure 4.38: Precision of the system with different trust decreasing factors.

agents do not provide opinions or advice, trust in them is never updated. What we want to study in this section is the effect of this parameter on the performance of the system. In order to do that, three different factors were analysed:

- $\chi = 1.0$: agents do not periodically decrease trust in their friends. All previous experiments have been performed with this value.
- $\chi = 0.999$: trust in reliable agents is slightly decreased at every cycle.
- $\chi = 0.99$: the size of the decrease is higher than in other simulations.

No important differences are expected in the results of these simulations. It is important to note that this decrease affects all the agents in the contact list in the same way. Moreover, if we decrease all the trust values equally, since the final recommendations are ponderated by trust in the other agents, the result is exactly the same. Only agents who never provide opinions or advice will be affected by this parameter; they will lose positions at the top of the contact list. Figures 4.38 to 4.44 show the results of the simulations with different trust decreasing factors.

Precision. Figure 4.38 shows the precision of these simulations. All the simulations exhibit a similar precision when agents are very selective (around $\omega = 0.9$). This leads us to believe that the most reliable agents always provide information

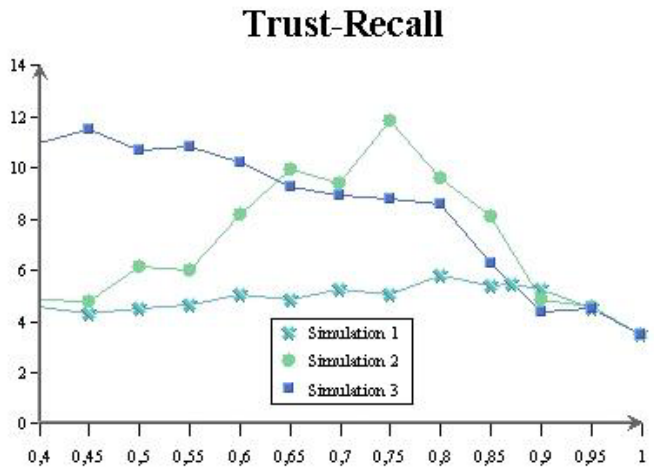
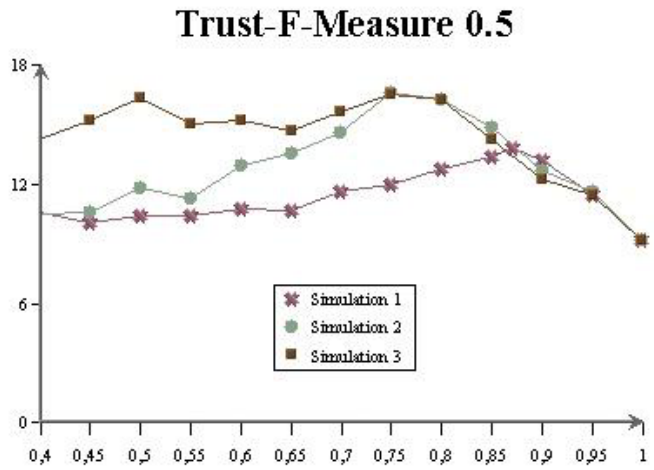


Figure 4.39: Recall of the system with different trust decreasing factors.

and, therefore, the trust decreasing factor does not affect them very much. When agents rely on many other agents, simulations present different precisions. The higher the trust decreasing factor, the higher the precision of the system. The trust decreasing factor penalises all the agents in the contact lists and, therefore, only agents providing useful information are taken into account. However, since we claim the most outstanding performance of the system is between $\omega = 0.85$ and $\omega = 0.9$, the difference of precision in the other trust thresholds is not relevant.

Recall. We obtain similar results when the recall of the system is analysed (see Figure 4.39). The recall in the various simulations is very similar when the trust thresholds are very high. However, simulations exhibit a very different recall when many agents are contained in the contact lists. When $\chi = 0.999$, recall outlines a bell shaped graph and the best recall is found when $\omega = 0.75$. When $\chi = 0.99$, the best recall is found when most of the agents are contained in their contact lists. Therefore, the higher the trust decreasing factor, the higher the recall when the trust threshold is at its lowest.

F-Measure. The f-measure corroborates the results obtained with precision and recall (see Figures 4.40 and 4.41). The f-measure in the interval $[\omega = 0.85, \omega = 0.9]$ is similar for all the simulations. However, when ω is low, the f-measure presents a better result when a trust decreasing factor is applied. Moreover, the higher the



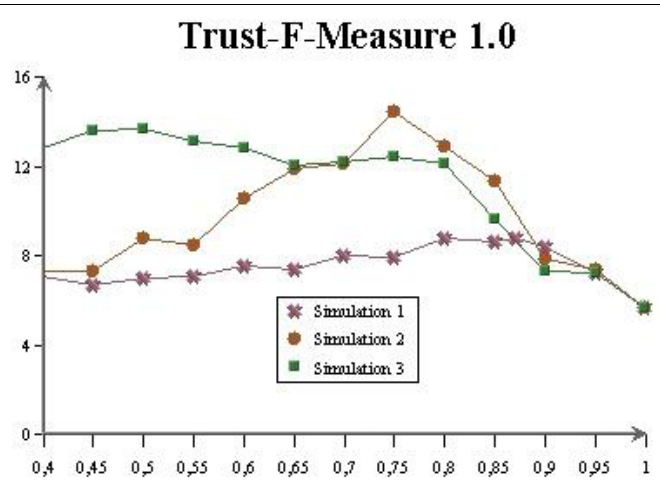
PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$
$Playing0$

Simulation1 - $\chi = 1.0$
Simulation2 - $\chi = 0.999$
Simulation3 - $\chi = 0.99$

$\varphi = 0.85$

Figure 4.40: F-Measure of the system with different trust decreasing factors when b=0.5.



PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$
$Playing0$

Simulation1 - $\chi = 1.0$
Simulation2 - $\chi = 0.999$
Simulation3 - $\chi = 0.99$

$\varphi = 0.85$

Figure 4.41: F-Measure of the system with different trust decreasing factors when b=1.0.

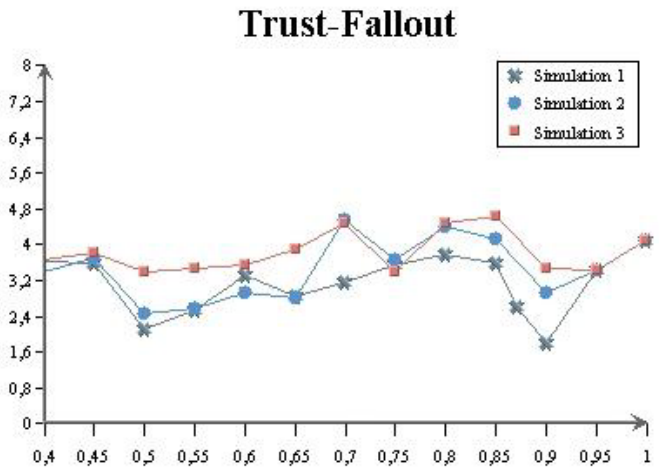


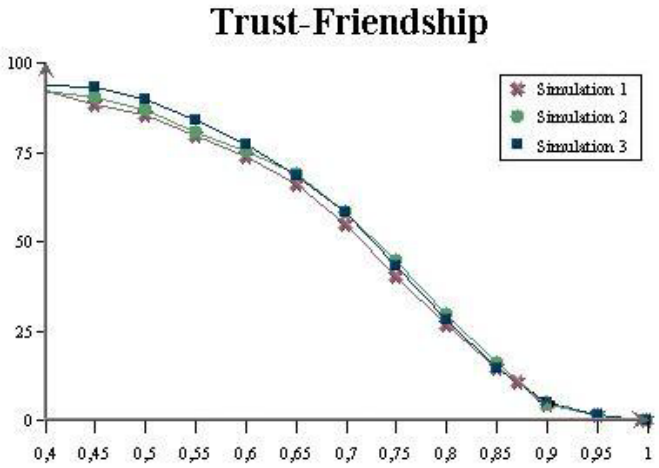
Figure 4.42: Fallout of the system with different trust decreasing factors.

importance of recall, the higher the f-measure when many agents are considered as reliable.

Fallout. No important differences are obtained when the fallout measure is analysed (see Figure 4.42). All the simulations exhibit a similar fallout ranging from 3 to 4, whatever the trust threshold.

Friendship. Simulations also exhibit a similar friendship (see Figure 4.43); that is, the trust decreasing factor does not affect the average percentage of agents contained in the contact lists over time. Reasonably, this parameter does not intervene in the incorporation of new friends into the contact lists. However, the number of agents intervening in the recommendation process; that is, the number of agents in the contact list above the trust threshold, is lower when there is a high trust decreasing factor. Trust values of useless agents have been decreased and their opinions and advice are not taken into account, especially when many agents are contained in the contact lists.

Diversity. The trust decreasing factor does not affect the diversity of the user profile at the end of the simulation (see Figure 4.44). However, the number of agents in a contact list does affect the diversity of the profile. The higher the



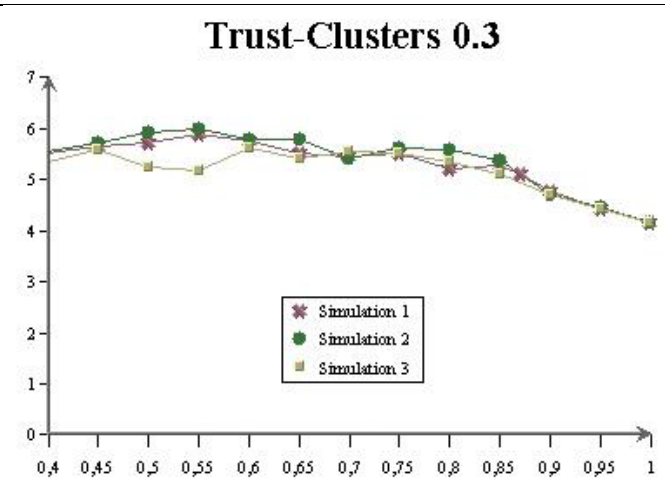
PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$
$Playing0$

Simulation1 - $\chi = 1.0$
Simulation2 - $\chi = 0.999$
Simulation3 - $\chi = 0.99$

$\varphi = 0.85$

Figure 4.43: Friendship of the system with different trust decreasing factors.



PARAMETERS

$\tau^+ = 0.7$ $\tau^- = 0.3$
$Playing0$

Simulation1 - $\chi = 1.0$
Simulation2 - $\chi = 0.999$
Simulation3 - $\chi = 0.99$

$\varphi = 0.85$

Figure 4.44: Diversity of the system with different trust decreasing factors.

number of friends, the more diverse the opinions and advice and, therefore, the more diverse the user profiles. However, the difference of diversity between interval $[\omega = 0.85, \omega = 0.9]$ and interval $[\omega = 0.5, \omega = 0.55]$ is not very significant, since only 0.8 clusters are lost.

Results on trust decreasing factors show that this parameter affects the performance of the system mainly when agents are less selective when making friends. When only the most reliable friends are contained in the contact lists, simulations show very similar results. Thus, when agents are more tolerant when adding new friends to contact lists, the trust decreasing factor causes an increase of precision and recall. This is mainly due to the fact that this parameter especially penalises agents who do not provide information and only useful friends are enquired during the recommendation process. Therefore, this parameter helps recommender agents in selecting which are the most useful reliable agents.

Trust Modifying Factors

The trust modifying factor φ controls the "acquaintance" of agents contained in the contact lists from the usefulness of their opinions and recommendations (see section 4.5.3). When an agent in the contact list gives good opinions or advice that results in a successful recommendation, the trust value of this agent is rewarded. Contrarily, if the resulting recommendation is badly evaluated by the user, the agent is penalised. In order to do that, equation 4.5 is used, where φ ponders the effect of success or failure on trust modification. In particular, four different trust modifying factors have been analysed:

- $\varphi = 0.0$: only the last opinion/advice is taken into account. Thus, a recommender agent relies blindly on other agents whose last opinion/advice resulted in success. Contrarily, agents who gave a badly opinion/advice in the last interaction are not believed.
- $\varphi = 0.5$: trust values are strongly affected by success or failure.
- $\varphi = 0.85$: trust values are slightly modified by success or failure.
- $\varphi = 1.0$: trust in other agents is never modified. Thus, the trust in agents remains regardless of the result of their opinions/advice.

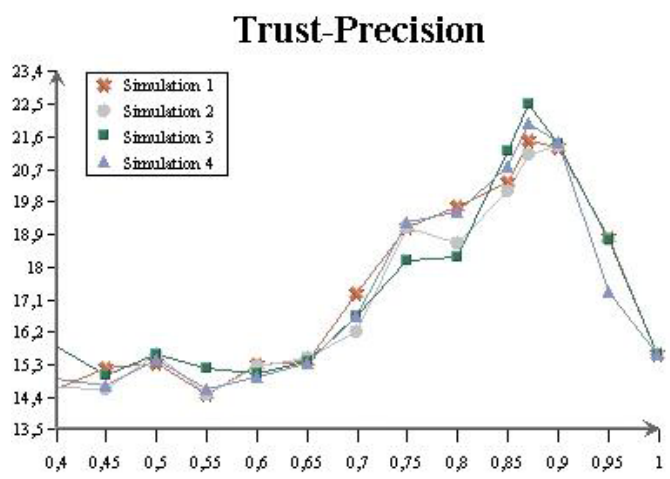


Figure 4.45: Precision of the system with different trust modifying factors.

The trust modifying factor is used to reward/penalise reliable agents who provide good/bad advice respectively. Therefore, when agents who do not provide useful information are contained in contact lists, depending on φ, equation 4.5 penalises them in order to improve the performance of the system. We expect that simulations with $\varphi = 1.0$; that is, when trust is not modified, give the worst performance, since agents who provide bad opinions/advice are always enquired. All other simulations update trust values, therefore, we cannot predict which performs best.

After analysing the simulations performed with the various trust modifying factors, we can see that all of them present a similar performance (see, for example, Figure 4.45). The reason for this is because the selected playing agents procedure gives a good turnover and only a very few agents of the contact lists give bad opinions/advice. In this case, we cannot estimate the effect of φ on the performance of the system.

The playing item set used in these simulations, *Playing*0, proves to be the one that best selects reliable friends (see section 4.7.4). In order to exhibit the φ effect, we have repeated the simulations with a playing item set that selects inappropriate friends. *Playing*5 selects reliable agents based on similar opinions on three items that the user has evaluated with an undeterminate value (around 0.5). These items are not relevant to users, thus, *Playing*5 does not provide the best set of friends.

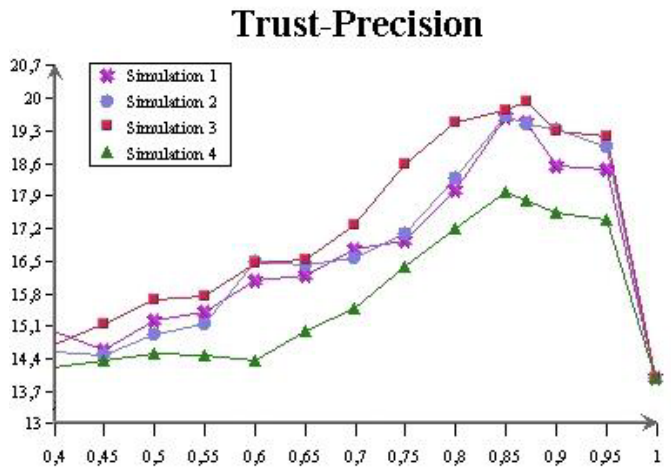


Figure 4.46: Precision of the system with different trust modifying factors II.

As explained in section 4.7.1, the *Training*3 initial item set has been used in all the simulations since it proved to be the best. This set consists of the three best and worst evaluated items by the user. If we want to select friends by means of *Playing*5, we need three items with an indeterminate assessment in the initial user profiles. Therefore, we have designed *Training*4, which incorporates the three items with the most indeterminate evaluation in *Training*3.

The results of the simulations with *Training*4, *Playing*5 and different trust modifying factors are shown in Figures 4.46 to 4.51.

Precision. Figure 4.46 shows the precision of the simulations with these parameters. We can see that the precision is lower than in the simulations with *Training*3 and *Playing*0 (compare Figures 4.45 and 4.46), but now the effect of the different trust modifying factors is appreciable. All the simulations exhibit a higher precision when ω is confined in the [0.8-0.95] interval. As expected, the simulation with the lowest precision is when $\varphi = 1.0$; that is, when no adaptation is performed. Agents maintain their reliance on their friends regardless of the result of their opinions/advice. The other simulations perform similarly (especially between $\omega = 0.85$ and $\omega = 0.95$), although precision is slightly better when $\varphi = 0.85$. Thus, in our experiments, a slow adaptation of trust values performs best. The high precision obtained when only the success of the last opinion/advice is taken into account is

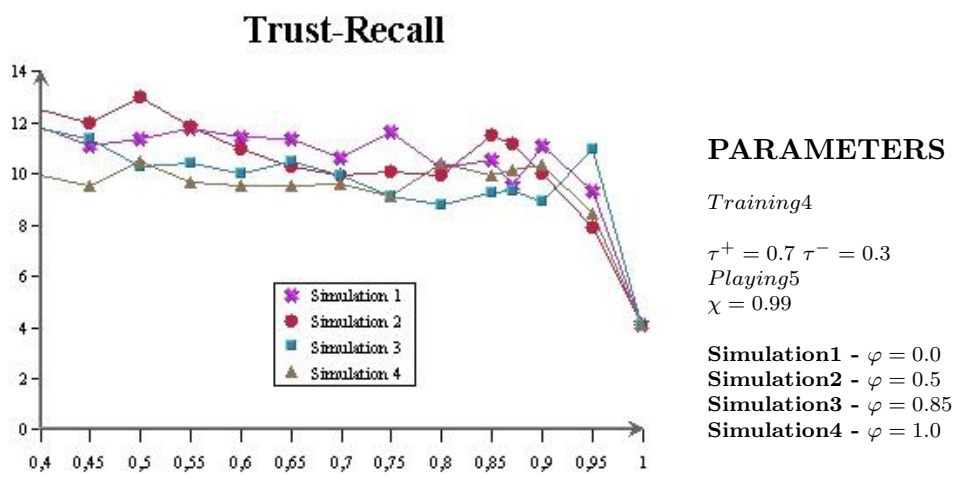
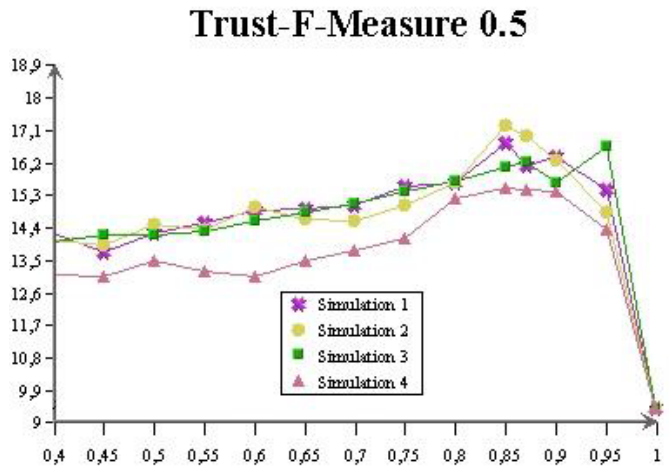


Figure 4.47: Recall of the system with different trust modifying factors.

very interesting. An adaptation of taking into account only the last result is considerably better than maintaining static contact lists over time. This result proves the importance of a continual validation of trust values.

Recall. Figure 4.47 shows the recall of the system when various trust modifying factors are analysed. All the simulations exhibit the highest recall when many agents are contained in the contact lists (around $\omega = 0.5$) and the lowest when no collaboration is performed ($\omega = 1.0$). No important recall differences can be appreciated in these simulations, thus, the trust modifying factor does not affect the recall of the system.

F-Measure. When precision and recall are combined, the results corroborate the fact that trust has to be updated according to the outcomes. When precision and recall are equally weighted (see Figure 4.49), the simulation with no adaptation is the one with the worst result, although a similar f-measure is observed between $\omega = 0.8$ and $\omega = 0.9$. Since the recall of the simulation is lower in this interval when $\varphi = 0.85$, the f-measure is also slightly lower. When precision is twice as important as recall (see Figure 4.48), the f-measure is the lowest regardless of ω when $\varphi = 1.0$. The other simulations exhibit a very similar result in both graphs.



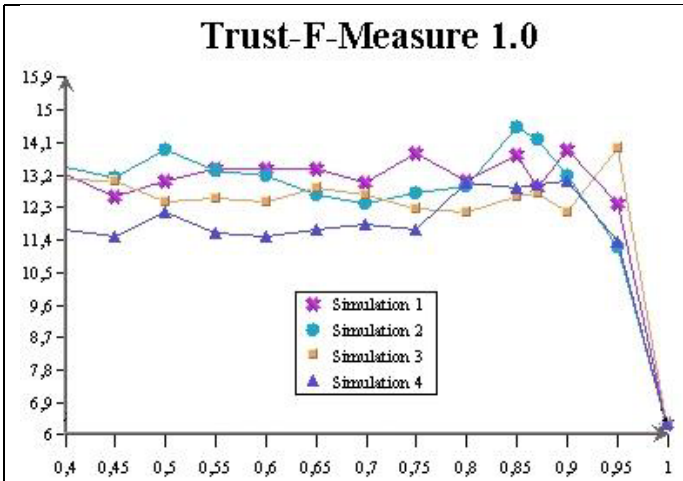
PARAMETERS

*Training*4

$\tau^+ = 0.7$ $\tau^- = 0.3$
*Playing*5
$\chi = 0.99$

Simulation1 - $\varphi = 0.0$
Simulation2 - $\varphi = 0.5$
Simulation3 - $\varphi = 0.85$
Simulation4 - $\varphi = 1.0$

Figure 4.48: F-Measure of the system with different trust modifying factors when b=0.5.



PARAMETERS

*Training*4

$\tau^+ = 0.7$ $\tau^- = 0.3$
*Playing*5
$\chi = 0.99$

Simulation1 - $\varphi = 0.0$
Simulation2 - $\varphi = 0.5$
Simulation3 - $\varphi = 0.85$
Simulation4 - $\varphi = 1.0$

Figure 4.49: F-Measure of the system with different trust modifying factors when b=1.0.

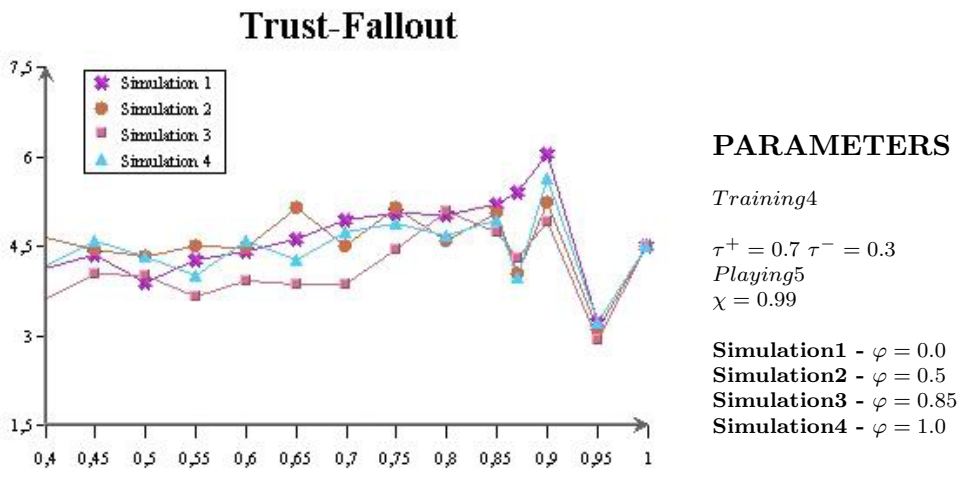


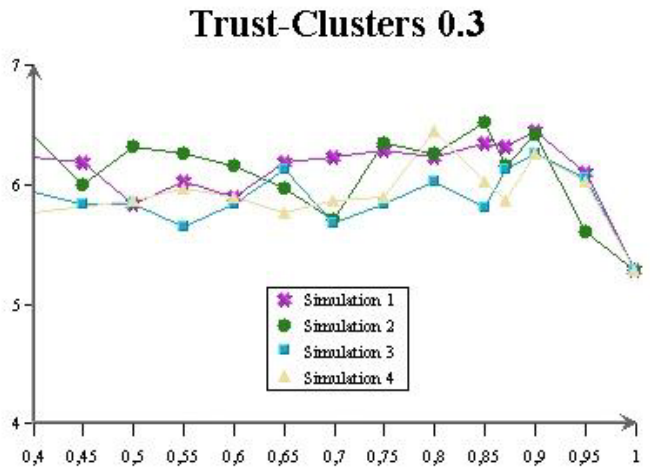
Figure 4.50: Fallout of the system with different trust modifying factors.

Fallout. The simulations with different trust modifying factors present similar fallout, although little difference can be observed (see Figure 4.50). The simulation with the lowest fallout between $\omega = 0.4$ and $\omega = 0.75$ is clearly when $\varphi = 0.85$. This result corroborates that, in this domain, this simulation performs better than the others.

Friendship. The trust modifying factor does not affect the selection of new friends. Therefore, all the simulations have the same friendship.

Diversity. No important differences can be seen in the diversity of the final case bases due to the correlation between recall and diversity (see Figure 4.51). A similar number of recommendations results in a similar diversity in the final case bases.

Summing up, the results of the simulations with different trust modifying factors have shown that it is very important to update the trust values in other agents. The size of φ is strongly related to the speed in which the user interests change over time. Reasonably, if users change their preferences constantly, trust values should be updated frequently in order to better adapt contact lists. Contrarily, if the interests of the users remain the same as time passes, a good/bad recommendation should not affect trust in other agents very much. Since it is very difficult to know the speed in



PARAMETERS

*Training*4

$\tau^+ = 0.7$ $\tau^- = 0.3$
*Playing*5
$\chi = 0.99$

Simulation1 - $\varphi = 0.0$
Simulation2 - $\varphi = 0.5$
Simulation3 - $\varphi = 0.85$
Simulation4 - $\varphi = 1.0$

Figure 4.51: Diversity of the system with different trust modifying factors.

which the interests of the users of the other agents change, we assert that the trust modifying factor has to be determined according to the domain of recommendation. In domains where users use to change their preferences quickly (for example in the supermarket purchases), a low trust modifying factor provides fast adaptation to the outcomes. However, in domains where the interests of the users persist for a long time (for example, the domain in which the experiments of this thesis has been performed: the restaurant domain), a higher trust modifying factor is suitable.

4.8 Chapter Conclusions

When recommender systems want to take advantage of the collaborative world, privacy issues arise. Recently, the protection of personal data has gained a great deal of importance with the disproportionate growth of Internet. The typical collaborative filtering method, which has proved very efficient, does not ensure the privacy of the user's information. In this chapter we have proposed a new method to deal with the trade-off between privacy and collaboration. We propose recommender agents that encapsulate personal data and collaborate by means of the opinion-based filtering method (OBF) and the collaborative filtering method based on trust (CFT). The OBF method allows users to exchange opinions that represent the user interests in a certain item. By means of these opinions, agents can interact and find other agents

on which to rely. Reliability is expressed through a trust value. A model that contemplates trust dynamics has been proposed as well. Thanks to this trust model, the CFT method can be applied. Agents obtain collaborative recommendations from their reliable friends taking advantage of the collaborative world.

Therefore, only general information about the user interest is revealed. This is how we deal with the trade-off between privacy and collaboration.

In order to study the OBF and the CFT methods and the effect of the different parameters involved in the collaboration process, a new evaluation method has also been proposed in this chapter: the profile discovering procedure with collaborations simulates a MAS of recommender agents exchanging opinions and advice.

The experimental results show that the methods proposed improve the results obtained with the typical content-based and collaborative filtering methods. However, the OBF and the CFT methods are very sensitive to the number of reliable agents contacted in the collaboration process. The outstanding performance is obtained when only agents with a high trust value are kept in their contact lists.

After presenting the results of our proposal from a global point of view, we analysed how different doubt thresholds, different playing item sets and different trust adaptation parameters affect the performance of the system. After this exhaustive analysis, we can conclude:

- If agents recommend only new items to the user with a high certainty of being accepted (high doubt threshold), the precision of the system is extremely high, although only very few items are recommended. When the certainty of the recommendation is low (low doubt threshold), recall is very high, but precision decreases considerably. Therefore, the doubt thresholds have to be selected depending on the purposes of the recommender system.

- The set of items to be compared in order to make reliable friends conditions the results of the system. A small set of items is very selective and only a few agents are considered as reliable. If we compare all the items in the profile (the collaborative filtering approach), the contact lists contain more agents but results are worse. It is important to have a good criteria when selecting items to compare. The best results are obtained when only the three most preferred and the three most disliked items are presented to the user.

- A periodical decrease of the trust values produces a better performance of the system when many agents are considered as reliable. A suitable trust decreasing factor helps recommender agents to select which reliable agents are the most useful.

- It is very important to update the trust in other agents according to the outcomes. When agents give bad opinions/advice, their trust values should be decreased. On the contrary, a beneficial collaboration should be rewarded by increasing trust values. The size of such modifications is relative to the speed in which the user interests change over time.

Finally, we want to note that this new approach emphasises proactiveness of agents while preserving privacy. That is to say, when an agent does not have enough knowledge to decide about a recommendation, it will turn to other agents on the web in order to look for similar agents from whom to gather information.

Chapter 5

Conclusions and Future Work

This chapter presents the principal contributions of this thesis. Further work in certain subjects in which the research of this thesis can continue is also included. In conclusion, there is a list of related publications and prizes.

5.1 Contributions

This thesis has focussed on the study of recommender systems. In particular, we have proposed a recommender system consisting of collaborative recommender agents based on case-based reasoning (CBR) and trust.

The main contributions of this thesis work are:

1. **Taxonomy of recommender systems**: A thorough analysis of existing recommender systems has resulted in a survey of state-of-the-art recommender systems on the Internet. This work has been organised as a taxonomy, which classifies recommender systems into 8 general dimensions; five regarding profile generation and maintenance and three concerning profile exploitation. The taxonomy provides a comprehensive explanation of existing recommender systems which we hope will contribute towards progress in this area of research.

2. **CBR approach to recommender systems**: the CBR cycle has been redefined in order to perform the recommendation task. Assuming that the user has similar interests in similar items, the recommender system predicts the

user preferences in new items from the implicit/explicit interest given by the user in similar items.

3. **Forgetting mechanism for case-based profiles**: One of the main contributions of this thesis is a forgetting mechanism for case-based profiles. This mechanism is based on the drift attribute, a case attribute that controls the relevance and age of the cases. The drift attribute is a real value that is updated as time passes according to the results of the recommendation process. The experimental results have proved that the forgetting mechanism increases the precision of the system while reducing the number of cases in the case base. Thus, the proposed mechanism better adapts the case-based profile to the user and solves the utility problem.

4. **Collaboration with privacy**: Recommender systems sharply improve the quality of their results when information about other users is utilised when recommending to a given user. The collaborative filtering method has shown important results in this sense, although this method requires the revelation of personal information about the users. In order to maintain the privacy of the users' personal data, we have proposed a new mechanism of collaboration based on intelligent agents. Agents encapsulate the user profile and are in charge of recommending interesting items to the user. In order to deal with the trade-off between privacy and collaboration, recommender agents exchange opinions representing the user's general interests without revealing detailed information.

5. **Opinion-based filtering method**: Another contribution to recommender systems is a new multi-agent information filtering method called the opinion-based filtering method. This method consists of reinforcing the recommendation process with opinions from other agents. When recommender agents are not sure about a recommendation, they ask for an opinion on the given item to other agents. If the opinions are favourable, the item is recommended to the user. Experimental results have shown that this method improves the performance of a non-collaborative recommender system and maintains the privacy of personal information.

6. **Social model of trust for recommender agents**: Recommender agents consider other agents as personal entities on whom they can rely or not in the

collaboration process. Reliability is expressed through a trust value with which each agent labels its neighbours. Our contribution to the multi-agent systems is a social model of trust for recommender agents. This model comprehends the initialisation and evolution of trust; that is, trust dynamics. Thanks to the opinion-based filtering method, agents can exchange opinions and learn about each other in order to generate and update trust values. The proposed social model of trust makes personal agents more robust when collaborating with other agents of the system.

7. **Collaborative filtering method through trust**: When a recommender system is implemented as a distributed world of recommender agents, the typical collaborative filtering method cannot be applied. The fact that recommender agents encapsulate user profiles makes it impossible for direct comparison between them; the basis of the collaborative filtering method. However, thanks to the trust model generated with the opinion-based filtering method, agents know who will give them good advice. Thus, we also contribute to the multi-agent information filtering methods with an evolution of the collaborative filtering method based on our social trust model. This method improves the performance of the system sharply when only the most reliable agents are enquired. The collaborative filtering method through trust, together with the opinion-based filtering method, can be seen as an evolution of the existing methods due to the agent's theory (see Figure 5.1).

8. **Evaluation procedure for recommender systems**: In order to test the CBR approach to recommender systems and the forgetting mechanism for case-based profiles, we have provided a new methodology for measuring the performance of a recommender system. The profile discovering procedure simulates the exploitation of a recommender system based on real user profiles. Based on this procedure, we have developed a simulator in order to carry out repeatable and perfectly controlled experiments quickly. Furthermore, we have also designed a new evaluation methodology in order to test our collaborative recommender agents. This method is called the profile discovering procedure with collaboration and is an extension of the procedure used to evaluate non-collaborative recommender systems. We have also designed a simulator that emulates the recommendation process and the collaboration of recommender agents based on real user profiles. Our contribution is general enough to be

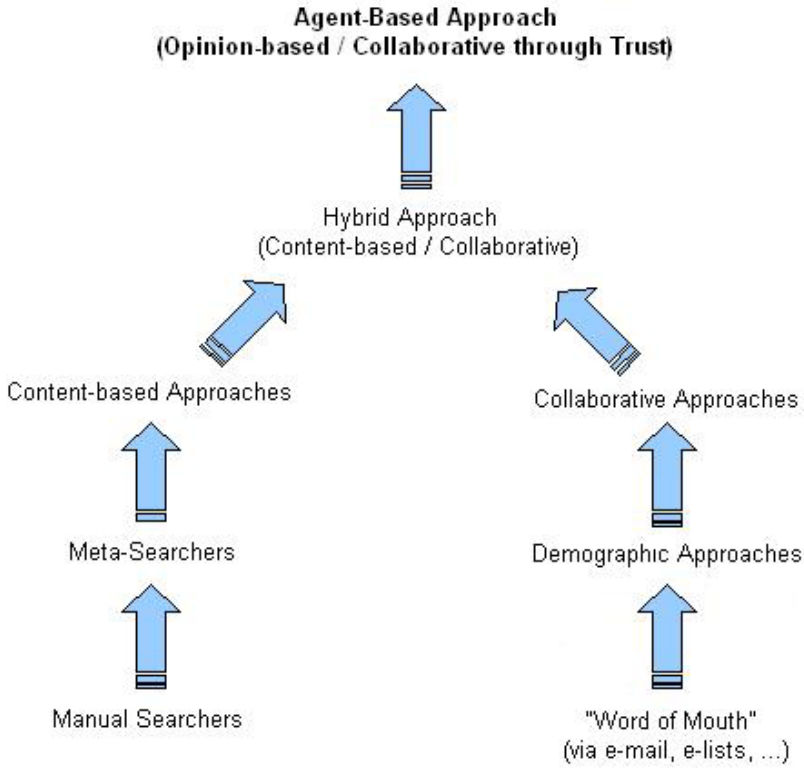


Figure 5.1: Evolution of Information Filtering Methods.

applied to other CBR/non-CBR recommender systems.

5.2 Future Work

The design of a recommender system involves the consideration of a wide range of questions. In addition to the different solutions which have been adopted and described in this thesis, many ideas have been proposed, discussed and finally rejected. On the other hand, other questions have remained as undeveloped ideas, which need to be analysed further and worked on in depth in future work.

Some subjects in which the research of this thesis can continue are:

- **Trust adaptation functions**: The adaptation of contact lists in our trust model according to the outcomes of the recommendation process have been proved useful in order to better collaborate. The functions selected in this thesis are an easy approach that modifies trust values according to the results of the recommendations provided by the enquired agents. Our first future goal is to study other trust adaption functions in order to improve agents acquaintance.

- **Social networks**: Contact lists represent the social relation an agent has with others. The representation of all the contact lists together results in a social network where nodes represent the recommender agents of the users and links the collaborative relations. A very important project in our future research is the study of such social networks. Social behaviours, network evolution, community formation and community structures are interesting aspects to be studied.

- **Agentification of existing recommender systems**: An important objective in the continuing research of this thesis is to test our proposals in different existing recommender systems in order to validate the outstanding performance shown in our experiments. In particular, we want to study the possibility of applying the proposed information filtering methods to existing recommender systems by means of agentification. For example, the Computer Science Department of the University of Bath (UK) is interested in integrating

our proposal of collaborative recommender agents into their bed and breakfast reservation system (B&B-Bath).

- **Apply the opinion-based filtering method to rescue**: Our research group recently started a project of rescue in coordination with other universities. The intention of this project is to promote research and development in the domain of disaster rescue at various levels. In particular, we are in charge of optimising the coordination among the different emergency services by means of multi-agent team work coordination, information infrastructures, personal digital assistants and decision support systems. Our aim is to study the performance of the opinion-based filtering method in such a crucial domain.

- **CBR multi-domain collaboration**: Certainly, the food that people buy in the supermarket reflects their interests in restaurants. A user who likes Italian restaurants will buy pasta at the supermarket. Thus, recommender systems should be able to predict the user interests in one domain from information about another one. Our idea is to study the possible collaboration among case-based profiles of different domains in order to improve the performance of a CBR recommender system.

- **Smart user models**: The idea is to create an adaptive user model that captures the evolution of the users regarding their emotions. Emotional Intelligence has been described as an important part of human decision making [Goleman 95]. It has been proved that, at a neurological level, emotions play a definitive role in the cognitive process [Joseph 01]. From our point of view then, a user model based on a set of objective and subjective characteristics, quantitatively and qualitatively measurable, is not enough to build systems aimed at supporting human decision-making. The emotional factors have to be added to the user model in order to define a Smart User Model.

5.3 Related Publications and Prizes

The motivation of this thesis has been published as a book chapter:

- de la Rosa J.Ll., del Acebo, E., López, B. and Montaner, M., "*From Physical Agents to Recommender Agents*". Intelligent Information Agents - The AgentLink Perspective. Edited by Matthias Klusch, Sonia Bergamaschi, Pete Edwards and Paolo Petta. Lecture Notes in Computer Science 2586. ISBN 3-540-00759-8. pp. 165-178. Springer. March, 2003.

The general study of recommender agents presented in chapter 1 has been published in an international journal:

- Montaner, M., López, B., de la Rosa, J. Ll., "*A Taxonomy of Recommender Agents on the Internet*", Artificial Intelligence Review, Kluwer Academic Publishers. Volume 19, Issue 4, pp. 285-330. June, 2003.

The proposals of this thesis have been presented and discussed in different international conferences:

- Montaner, M., López, B., de la Rosa, J. Ll., "*Developing Trust in Recommender Agents*". In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02). Cristiano Castelfranchi and W. Lewis Johnson (Eds). ACM Press. vol. 1, pp. 304-305. Bologna (Italy). 15-19 July, 2002.

- Montaner, M., López, B., de la Rosa, J. Ll., "*Improving Case Representation and Case Base Maintenance in Recommender Agents*". In Proceedings of the 6th European Conference on Case Based Reasoning (ECCBR'02). Susan Craw, Alun Preece (Eds.), Lecture Notes in AI Nº2416. Springer-Verlag. pp. 234-248. Aberdeen (Scotland). 4-7 September, 2002.

- Montaner, M., López, B., de la Rosa, J. Ll., "*Opinion-Based Filtering Through Trust*". In Proceedings of the 6th International Workshop on Cooperative Information Agents (CIA'02). Matthias Klusch, Sascha Ossowski and Onn Shehory (Eds.), Lecture Notes in AI Nº2446. Springer-Verlag Berlin Heidelberg, pp. 164-178, Madrid (Spain). 18-20 September, 2002.

- Montaner, M., López, B., del Acebo, E., Aciar, S., "*Towards a Collaborative Filtering Method in an Open World*". In Proceedings of the International Workshop of the AgentCities Third Information Day (iD3). Barcelona (Spain). 6-8 February, 2003.

The implementation of the recommender system proposed in this thesis has been awarded with two prizes:

- "*IRES: On the Integration of Restaurant Services*". Awarded with the Special Prize of the AgentCities Agent Technology Competition. Barcelona (Spain). 6-8 February, 2003.
- "*GenialChef*". Awarded with the Prize for the Best University Project of the E-TECH 2003. Girona (Spain). 3-5 April, 2003.

Finally, the results of this thesis presented in chapters 3 and 4 have been submitted to the international journals *Machine Learning* and *Autonomous Agents and Multi-Agent Systems* respectively.

Bibliography

[Aamodt 94] A. Aamodt and E. Plaza. *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*. volume 7, pages 39–59. AICom - Artificial Intelligence Communications, IOS Press, 1994.

[Aha 91] D. W. Aha, D. Kibler and M. K. Albert. *Instance-Based Learning Algorithms*. In Machine Learning, volume 6, pages 37–66, 1991.

[Aha 92] D. W. Aha. *Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms*. In International Journal of Man-Machine Studies, volume 36, pages 267–287, 1992.

[Amazon 03] Amazon. *http://www.amazon.com*, 2003.

[Armstrong 95] R. Armstrong, D. Freitag, T. Joachims and T. Mitchell. *Webwatcher: A learning apprentice for the world wide web*. In 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments, 1995.

[Asnicar 97] F. Asnicar and C. Tasso. *ifWeb: a Prototype of user models based intelligent agent for document filtering and navigation in the world wide web*. In Sixth International Conference on User Modeling. Chia Laguna, Sardinia, Italy, 1997.

[Balabanovic 97] M. Balabanovic and Y. Shoham. *Combining Content-Based and Collaborative Recommendation*. In Communications of the ACM, 1997.

[Basu 98] C. Basu, H. Hirsh and W. Cohen. *Recommendation as Classification: Using Social and Content-Based Information in Recommendation*. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98), pages 714–720. AAAI Press/MIT Press, 1998.

[Berney 99] B. Berney and E. Ferneley. *CASMIR: Information Retrieval Based on Collaborative User Profiling*. In Proceedings of the Forth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'99), pages 41–56. The Practical Application Company Ltd, Lancashire, 1999.

[Billsus 98] D. Billsus and M. J. Pazzani. *Learning Collaborative Information Filters*. In Proceedings of the International Conference on Machine Learning. Madison, Wisc: Morgan Kaufmann Publishers, 1998.

[Billsus 99] D. Billsus and M. J. Pazzani. *A Hybrid User Model for News Classification*. In Kay J. (ed.), UM99 User Modeling - Proceedings of the Seventh International Conference, pages 99–108. Wien, New York: Springer-Verlag, 1999.

[Boone 98] G. Boone. *Concept Features in Re:Agent, an Intelligent Email Agent*. In The Second International Conference on Autonomous Agents (Agents '98). Minneapolis/St. Paul, 1998.

[Bowbrick 00] S. Bowbrick. *Multiple Identities in the Online World*. In Proceedings of The First Annual Consult Hyperion Digital Identity Forum. London, 2000.

[Breese 98] J. Breese, D. Heckerman and C. Kadie. *G. F. Cooper and S. Moral, editors, Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*. In Kay J. (ed.), UM99 User Modeling - Proceedings of the Seventh International Conference, pages 43–52. Morgan Kaufmann, San Francisco, 1998.

[Brodley 93] C. E. Brodley. *Addressing the Selective Superiority Problem: Automatic Algorithm/Model Class Selection.* In Proceedings of the Tenth International Machine Learning Conference, pages 17–24. Amherst, MA, 1993.

[Buckley 95] C. Buckley and G. Salton. *Optimization of relevance feedback weights.* In Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 351–357. Fox, Ed, Ingwersen, Peter, and Fidel, Raya (Editors), 1995.

[Buckley 96] C. Buckley, A. Singhal, M. Mitra and G. Salton. *New Retrieval Approaches Using SMART: TREC 4.* In publishing of the Fourth Text Retrieval conference (TREC-4). NIST Special Publication, 1996.

[Burke 97] R. Burke, K. Hammond and B. Young. *The FindMe Approach to Assisted Browsing.* In IEEE Expert, volume 12(4), pages 32–40, 1997.

[Burke 00] R. Burke. *Semantic ratings and heuristic similarity for collaborative filtering.* In AAAI Workshop on Knowledge-based Electronic Markets 2000 (KBEM'00). Austin, TX, 2000.

[Canny 02a] J. Canny. *Collaborative Filtering with Privacy.* In IEEE Symposium on Security and Privacy, pages 45–57, 2002.

[Canny 02b] J. Canny. *Collaborative Filtering with Privacy via Factor Analysis.* In ACM Conference on Research and Development in Information Retrieval (SIGIR 2002), 2002.

[Carroll 87] J. Carroll and M. B. Rosson. *The Paradox of the Active User.* In Interfacing Thought: Cognitive Aspects of Human-Computer Interaction, pages 26–28. J. M. Carroll (Ed.), Cambridge, MA: MIT Press, 1987.

[Castelfranchi 98] C. Castelfranchi and R. Falcone. *Principles of Trust for MAS: Cognitive Anatomy, Social Importance, and Quantification.* In

Demazeau, Y. (ed.), Proceedings of the Third International Conference on Multi-Agent Systems, pages 72–79. IEEE Computer Society, Los Alamitos, 1998.

[Castelfranchi 01] C. Castelfranchi. *Information Agents: The Social Nature of Information and the Role of Trust*, 2001. Invited Contribution at Cooperative Information Agents V (CIA'01). Modena (Italy).

[Cayzer 02] S. Cayzer and U. Aickelin. *A Recommender System based on the Immune Network*. In Technical Report HPL-2002-1. HP Laboratories Bristol, 2002.

[CDNow 03] CDNow. *http://www.cdnow.com*, 2003.

[Chang 74] C.-L. Chang. *Finding Prototypes for nearest Neighbor Classifiers*. In IEEE Transactions on Computers, volume 23:11, pages 1179–1184, 1974.

[Chatterjee 98] P. Chatterjee, D. L. Hoffman and T. P. Novak. *Modeling the Clickstream: Implications for Web-Based Advertising Efforts*. In Working Paper. Vandetbilt University, 1998.

[Chen 98] L. Chen and K. Sycara. *WebMate: A Personal Agent for Browsing and Searching*. In Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems, AGENTS '98, pages 132–139. ACM, 1998.

[Chen 00] Z. Chen, X. Meng, B. Zhu and R. Fowler. *Websail: From on-line learning to web search*. In Proceedings of the 2000 International Conference on Web Information Systems Engineering, 2000.

[Clark 89] P. Clark and T. Niblett. *The CN2 Induction Algorithm*. In Machine Learning, volume 3, pages 261–83. Kluwer Academic Publishers, The Netherlands, 1989.

[Cohen 95] W. Cohen. *Fast Effective Rule Induction*. In Proceedings of the 12th International Machine Learning Conference (ML95), pages 115–123. San Francisco, CA: Morgan Kaufmann, 1995.

[Cohen 99] W. Cohen and Y. Singer. *A Simple, Fast, and Effective Rule Learner*. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), pages 335–342, 1999.

[Cooley 99] R. Cooley, P. N. Tan and J. Srivastava. *Websift: the Web site information filter system*. In Proceedings of the 1999 KDD Workshop on Web Mining. San Diego, CA: Springer-Verlag, 1999.

[Cost 93] S. Cost and S. Salzberg. *A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features*. In Machine Learning, volume 10, pages 57–78, 1993.

[Cunningham 01] P. Cunningham, R. Bergmann, S. Schmitt, R. Traphoner, S. Breen and B. Smyth. *WEBSELL: Intelligent Sales Assistants for the World Wide Web*. In e-Business and e-Work 2001 (e-2001), 2001.

[Delgado 01] J. Delgado and N. Ishii. *Multi-Agent Learning in Recommender Systems for Information Filtering on the Internet*. In International Journal of Cooperative Information Systems, volume 10:1-2, pages 81–100, 2001.

[Dickinson 03] I. Dickinson, D. Reynolds, D. Banks, S. Cayzer and P. Vora. *User Profiling with Privacy: A Framework for Adaptative Information Agents*. In Intelligent Information Agents - The AgentLink Perspective, pages 123–151. LNAI 2586, 2003.

[Domingos 95] P. Domingos. *Rule Induction and Instance-Based Learning: A Unified Approach*. In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95). Morgan Kaufman, 1995.

[Domingos 96] P. Domingos. *Unifying Instance-Based and Rule-Based Induction*. In Machine Learning, volume 24, pages 141–168, 1996.

[Duda 73] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. In John Wiley & Sons, New York, ISBN-0471223611, 1973.

[Elofson 98] G. Elofson. *Developing Trust with Intelligent Agents: An Exploratory Study.* In Proceedings of the First International Workshop on Trust, pages 125–139, 1998.

[Falkman 00] G. Falkman. *Similarity Measures for Structured Representations: A Definitional Approach.* In Blanzieri, E. and Portinale, L., eds., Advances in Case-Based Reasoning. Proceedings of the 5th European Workshop, EWCBR 2000, Trento, Italy, volume 1898, pages 380–392. Springer-Verlag, Berlin, 2000.

[FIPA 03] FIPA. *http://www.fipa.org/specifications/index.html*, 2003.

[Gambetta 90] D. Gambetta. *Can We Trust Trust?* In Trust: Making and Breaking Cooperative Relations, pages 213–237. Gambetta, D (editor). Basil Blackwell. Oxford, 1990.

[Gates 72] G. W. Gates. *The Reduced Nearest Neighbor Rule.* In IEEE Transactions on Information Theory, volume IT-18-3, pages 431–433, 1972.

[Glance 98] N. Glance, D. Arregui and M. Dardenne. *Knowledge Pump: Supporting the Flow and Use of Knowledge.* In Information Technology for Knowledge Management, pages 35–45. Eds. U. Borghoff and R. Pareschi, New York: Springer-Verlag, 1998.

[Gnutella 03] Gnutella. *http://gnutella.wego.com*, 2003.

[Goker 00] M. H. Goker and C. A. Thompson. *The Adaptive Place Advisor: A Conversational Recommendation System.* In Proceedings of the 8th German Workshop on Case Based Reasoning, Lammerbuckel, Germany., 2000.

[Goldberg 92] D. Goldberg, D. Nichols, B. M. Oki and D. Terry. *Using Collaborative Filtering to Weave an Information Tapestry.* In Communications of the ACM, volume 35, pages 61–70, 1992.

[Goleman 95] D. Goleman. Emotional intelligence. ISBN 0-553-09503-X, Bantam Books, 1995.

[Good 99] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker and J. Riedl. *Combining collaborative filtering with personal agents for better recommendations.* In Proceedings of AAAI, volume 35, pages 439–446. AAAI Press, 1999.

[Greening 97] D. Greening. *Building Consumer Trust with Accurate Product Recommendations.* In Likeminds White Paper LMWSWP-210-6966, 1997.

[Hart 99] P. E. Hart. *The Condensed Nearest Neighbor Rule.* In IEEE Transactions on Information Theory, volume 14, pages 515–516, 1999.

[Hayes 99] C. Hayes and P. Cunningham. *Smart Radio - a proposal.* In Trinity College Dublin, Computer Science, Technical Report, TCD-CS-1999-24, 1999.

[Hayes 00] C. Hayes and P. Cunningham. *Smart Radio: Building Music Radio on the Fly.* In Proceedings of Expert Systems 2000 (ES2000). Cambridge, UK, 2000.

[Hayes 01] C. Hayes, P. Cunningham and B. Smyth. *A Case-Based Reasoning View of Automated Collaborative Filtering.* In Trinity College Dublin, Computer Science, Technical Report, TCD-CS-2001-09, 2001.

[Herlocker 99] J. Herlocker, J. Konstan, A. Borchers and J. Riedl. *An Algorithmic Framework for Performing Collaborative Filtering.* In Proceedings of the 1999 Conference on Research and Development in Information Retrieval, 1999.

[Herlocker 00] J. Herlocker, J. Konstan and J. Riedl. *Explaining Collaborative Filtering Recommendations.* In Proceedings of ACM 2000 Conference on Computer Supported Cooperative Work, 2000.

[Hill 95] W. Hill, L. Stead, M. Rosenstein and G. Furnas. *Recommending and evaluating choices in a virtual community of use.* In

Proceedings of the Conference on Human Factors in Computing Systems (CHI'95), pages 194–201. Denver, CO, ACM Press, 1995.

[Hofmann 99] T. Hofmann and J. Puzicha. *Latent Class Models for Collaborative Filtering.* In the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99), pages 688–693. Stockholm, ISBN 1-55860-613-0, 1999.

[Holte 96] R. C. Holte and N. Y. Yan. *Inferring what a user is not interested in.* In AAAI Spring Symp. on Machine Learning in Information Access. Stanford, USA, 1996.

[Huberman 96] B. Huberman and M. Kaminsky. *Beehive: A System for Cooperative Filtering and Sharing of Information.* In Technical report, Dynamics os Computation Group. Xerox, Palo Alto Research Center, Palo Alto, CA, 1996.

[Jennings 93] A. Jennings and H. Higuchi. *A User Model Neural Network for a Personal News Service.* In User Modeling and User-Adapted Interaction, volume 3, pages 1–25, 1993.

[Jensen 96] F. V. Jensen. *An Introduction to Bayesian Networks.* New York: Springer, 1996.

[Joachims 97] T. Joachims, D. Freitag and T. Mitchell. *WebWatcher: A Tour Guide for the World Wide Web.* In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, pages 770–775. Nagoya, Japan, 1997.

[Jonker 99] C. M. Jonker and J. Treur. *Formal Analysis of Models for the Dynamics of Trust based on Experiences.* In F. J. Garijo, M. Boman (eds.), Multi-Agent System Engineering, Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99. Lecture Notes in AI, volume 1647, pages 221–232. Springer Verlag, Berlin, 1999.

[Joseph 01] S. Joseph. *AI.* In J@pan-Inc, n. 25, 2001.

[Kamba 95] T. Kamba, K. Bharat and M. C. Albers. *The Krakatoa Chronicle–An Interactive, Personalized, Newspaper on the Web.* In Proceedings of the Fourth International World Wide Web Conference, pages 159–170, 1995.

[Klusch 01] M. Klusch. *Information Agent Technology for the Internet: A Survey.* In Journal on Data and Knowledge Engineering, Special Issue on Intelligent Information Integration, volume 36:6. D. Fensel (Ed.), Elsevier Science, 2001.

[Kobsa 01] A. Kobsa, J. Koenemann and W. Pohl. *Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships.* In The Knowledge Engineering Review, forthcoming, 2001.

[Konstan 97] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon and J. Riedl. *GroupLens: Applying Collaborative Filtering to Usenet News.* In Communications of the ACM, volume 40, pages 77–87, 1997.

[Koychev 00] I. Koychev. *Gradual Forgetting for Adaptation to Concept Drift.* In Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning, 2000.

[Krulwich 95] B. Krulwich and C. Burkey. *ContactFinder: Extracting indications of expertise and answering questions with referrals.* In The working Notes of the 1995 Fall Symposium on Intelligent Knowledge Navigation and retrieval, pages 85–91. Technical Report FS-95-03, The AAAI Press, 1995.

[Krulwich 96] B. Krulwich and C. Burkey. *Learning user information interests through extraction of semantically significant phrases.* In Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access. Stanford, CA, 1996.

[Krulwich 97] B. Krulwich. *LIFESTYLE FINDER: Intelligent User Profiling Using Large-Scale Demographic Data.* In AI Magazine, volume 18:2, pages 37–45, 1997.

[Lang 95] K. Lang. *NewsWeeder: Learning to filter news.* In Proceedings of the Twelfth International Conference on Machine Learning, pages 331–339. Lake Tahoe, CA, 1995.

[Leake 98] D. B. Leake and D. C. Wilson. *Categorizing Case-Base Maintenance: Dimensions and Directions.* In Advances in Case-Based Reasoning: 4th European Workshop, EWCBR-98, Dublin (Ireland), 1998.

[Lewis 94] D. D. Lewis and W. A. Gale. *A sequential algorithm for training text classifiers.* In Proceedings of SIGIR94, 17th ACM International Conference on Research and Development in Information Retrieval, pages 3–12, 1994.

[Lieberman 95] H. Lieberman. *Letizia: An Agent That Assists Web Browsing.* In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95), pages 924–929, 1995.

[Lieberman 99] H. Lieberman, N. W. Van Dyke and A. S. Vivacqua. *Let's Browse: A Collaborative Web Browsing Agent.* In Proceedings of International Conference on Intelligent User Interfaces, pages 924–929, 1999.

[Macaya 02] D. Macaya, J. Meléndez and J. Colomer. *Case Based approach for supervision. Application to PID controllers.* In Proceedings of the 15th-IFAC World Congress. Barcelona (Spain), 2002.

[Maes 94] P. Maes. *Agents that reduce work and information overload.* In Communications of the ACM, volume 37:7, pages 30–40, 1994.

[Maloof 00] M. A. Maloof and R. S. Michalski. *Selecting examples for partial memory learning.* In Machine Learning, volume 41, pages 27–52, 2000.

[March 94] S. P. March. *Formalising Trust as a Computational Concept.* In Phd Thesis, Department of Computing Science and Mathematics, University of Stirling, 1994.

[Meléndez 01] J. Meléndez, J. Colomer and D. Macaya. *Case-Based Reasoning Methodology For Supervision.* In Proceeding of the European Control Conference (ECC2001). Oporto (Portugal), 2001.

[Meléndez 03] J. Meléndez, J. J. Mora, D. Llanos, M. Ruiz, J. Colomer, J. Sánchez and X. Corbella. *Classification of short duration faults (voltage sags) in transmission and distribution power systems.* In To be published in Proceedings of the European Control Conference (ECC'03). University of Cambridge (UK), 2003.

[Minio 96] M. Minio and C. Tasso. *User Modeling for Information Filtering on INTERNET Services: Exploiting an Extended Version of the UMT Shell.* In UM96 Workshop on User Modeling for Information Filtering on the WWW. Kailua-Kona, Hawaii, USA, 1996.

[Mitchell 85] T. M. Mitchell, S. Mahadevan and L. Steinberg. *LEAP: A learning apprentice for VLSI design.* In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, pages 573–580. Los Altos, California. Morgan Kaufmann, 1985.

[Mitchell 94] T. Mitchell, R. Caruana, D. Freitag, J. McDermott and D. Zabowski. *Experience with a Learning Personal Assistant.* In Communications of the ACM, volume 37:7, pages 81–91, 1994.

[Mladenic 96] D. Mladenic. *Personal WebWatcher: Implementation and Design.* In Technical Report IJS-DP-7472, Department of Intelligent Systems, J.Stefan Institute, Slovenia, 1996.

[Mobasher 00] B. Mobasher, R. Cooley and J. Srivastava. *Automatic Personalization Based on Web Usage Mining.* In Communications of the ACM, volume 43:8, 2000.

[Morita 94] M. Morita and Y. Shinoda. *Information filtering based on user behaviour analysis and best match text retrieval.* In Proceedings of the 17th ACM Annual International Conference on Research and Development in Information Retrieval (SIGIR'94), pages 272–81. Dublin, Ireland, Springer-Verlag, 1994.

[Moukas 97] A. Moukas. *Amalthaea: Information Filtering and Discovery using a Multiagent Evolving System.* In Journal of Applied AI, volume 11:5, pages 437–457. Dublin, Ireland, Springer-Verlag, 1997.

[Nichols 97] D. M. Nichols. *Implicit Rating and Filtering.* In Proceedings of 5 th DELOS Workshop on Filtering and Collaborative Filtering, pages 31–36, 1997.

[Orwant 95] L. J. Orwant. *Heterogeneous Learning in the Doppelganger User Modelling System.* In User Modelling and User Adapted Interaction, volume 4:2, pages 107–130, 1995.

[Pazzani 96] M. Pazzani, J. Muramatsu and D. Billsus. *Syskill & Webert: Identifying interesting web sites.* In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pages 54–61, 1996.

[Pazzani 97] M. Pazzani and D. Billsus. *Learning and Revising User profiles: The Identification of Interesting Web Sites.* In Machine Learning, volume 27, pages 313–331. Kluwer Academic Publishers, 1997.

[Pazzani 99] M. Pazzani. *A Framework for Collaborative, ContentBased and Demographic Filtering.* In Artificial Intelligence Review, 1999.

[Potter 88] G. Potter and R. Trueblood. *Traditional, Semantic, and Hyper-Semantic Approaches to Data Modeling.* In IEEE Computer, volume 21:6, pages 53–63, 1988.

[Pous 03] C. Pous, J. Colomer, J. Meléndez and J. L. de la Rosa. *Case Base management for Analog Circuits Diagnosis Improvement.* In To be published in Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR03). Trondheim (Norway), 2003.

[Pretschner 99] A. Pretschner and S. Gauch. *Ontology Based Personalized Search.* In Proc. 11th IEEE Intl. Conf. on Tools with Artificial Intelligence (ICTAI'99), pages 391–398, 1999.

[Quinlan 83] J. R. Quinlan. *Learning efficient classification procedures and their application to chess end games*. In Machine learning: an artificial intelligence approach, pages 463–482. edited by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, 1983.

[Quinlan 94] J. R. Quinlan. *The Minimum Description Length Principle and Categorical Theories*. In Cohen W.W. and Hirsh H.(eds.), Machine Learning: Proceedings of the Eleventh International Conference (ML94). Morgan Kaufmann, San Mateo, CA, 1994.

[Resnick 94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl. *Grouplens: An open architecture for collaborative filtering of netnews*. In Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work, Sharing Information and Creating Meaning, pages 175–186, 1994.

[Resnick 97] P. Resnick and H. Varian. *Recommender Systems*. In Communications of the ACM, pages 56–58, 1997.

[Rich 79] E. Rich. *User Modeling via Stereotypes*. In Cognitive Science, volume 3, pages 329–354, 1979.

[Riordan 95] A. Riordan and H. Sorensen. *An intelligent agent for high-precision information filtering*. In Proceedings of the CIKM-95 Conference, 1995.

[Ritter 75] G. Ritter, H. Woodruff, S. Lowry and T. Isenhour. *An Algorithm for a Selective Nearest Neighbor Decision Rule*. In IEEE Transactions on Information Theory, volume 21-6, pages 665–669, 1975.

[Sabater 00] J. Sabater and C. Sierra. *REGRET: A reputation model for gregarious societies*. In Research Report. Institut d'Investigació i Intel.ligència Artificial, 2000.

[Sakagami 97] H. Sakagami, T. Kamba and Y. Koseki. *Learning personal preferences on online newspaper articles for user behaviors*. In Proc. 6th Int. World Wide Web Conference, pages 291–300, 1997.

[Salton 83] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Publishing Company, New York, NY, 1983.

[Salton 88] G. Salton and C. Buckley. *Term-Weighting Approaches in Automatic Text Retrieval*. In Information Processing and Management, volume 24:5, pages 513–523, 1988.

[Salton 90] G. Salton and C. Buckley. *Improving Retrieval Performance by Relevance Feedback*. In Spark Jones and Willet, (Eds.) Readings in Information Retrieval, volume 24:5, pages 513–523. San Francisco, CA: Morgan Kauffman, 1990.

[Sanguesa 00] R. Sanguesa, U. Cortés and B. Faltings. *W9. Workshop on Recommender Systems*. In Autonomous Agents, 2000. Barcelona, Spain, 2000.

[Sarwar 98] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller and J. Riedl. *Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System*. In Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'98), pages 345–354, 1998.

[Sarwar 00] B. M. Sarwar, G. Karypis, J. A. Konstan and J. Riedl. *Analysis of Recommender Algorithms for E-Commerce*. In ACM E-Commerce 2000 Conference, 2000.

[Schafer 01] J. B. Schafer, J. Konstan and J. Riedl. *Electronic Commerce Recommender Applications*. In Journal of Data Mining and Knowledge Discovery, volume 5, pages 115–152, 2001.

[Schillo 99] M. Schillo and P. Funk. *Who can you Trust: Dealing with Deception*. In Proceedings of the Workshop Deception, Fraud and Trust in Agent Societies at the Autonomous Agents Conference, pages 95–106, 1999.

[Schillo 00] M. Schillo, P. Funk and M. Rovatsos. *Using trust for detecting deceitful agents in artificial societites.* In Applied Artificial Intelligence, Special Issue on Trust, Deception and Fraud in Agent Societies, 2000.

[Schwab 00] I. Schwab, A. Kobsa and I. Koychev. *Learning about Users from Observation.* In AAAI 2000 Spring Symposium: Adaptive User Interface, 2000.

[Schwab 01] I. Schwab, A. Kobsa and I. Koychev. *Learning User's interests through Positive Examples Using Content Analysis and Collaborative Filtering*, 2001. Submitted.

[Shardanand 94] U. Shardanand. *Social Information Filtering for Music Recommendation.* In MIT EECS M. Eng. thesis, also TR-94-04, Learning and Common Sense Group, MIT Media Laboratory, 1994.

[Shardanand 95] U. Shardanand and P. Maes. *Social information filtering: algorithms for automating 'word of mouth´.* In Conf. proc. on Human factors in computing systems (CHI'95), pages 210–217, 1995.

[Sheth 93] B. Sheth and P. Maes. *Evolving agents for personalitzed information filtering.* In Proceedings of the Ninth Conferece on Artificial Intelligence for Applications. IEEE Computer Society Press, 1993.

[Sheth 94] B. Sheth. *A Learning Approach to Personalized Information Filtering.* In M.S. Thesis, Massachusetts Institute of Technology, 1994.

[Sneath 73] P. Sneath and R. Sokal. *Numerical Taxonomy. The principles and practice of numerical classification.* In Freeman, San Francisco, 1973.

[Sorensen 95] H. Sorensen and M. McElligot. *PSUN: A Profiling System for Usenet News.* In CKIM '95 Workshop on Intelligent Information Agents, 1995.

[Sorensen 97] H. Sorensen, A. O. Riordan and C. O. Riordan. *Profiling with the INFOrmer Text Filtering Agent.* In Journal of Universal Computer Science, volume 3:8, pages 988–1006, 1997.

[Steels 97] L. Steels and P. Vogt. *Grounding adaptive language games in robotic agents.* In Proceedings of the Fourth European Conference on Artificial Life, pages 473–484, 1997.

[Stefani 98] A. Stefani and C. Strappavara. *Personalizing access to web sites: The SiteIF project.* In Proc. 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98, 1998.

[Terveen 01] L. G. Terveen and W. Hill. *Beyond Recommender Systems: Helping People Help Each Other.* In Carroll, J. (ed.), HCI in the New Millennium. Addison Wesley, 2001.

[Tomek 76] I. Tomek. *An Experiment with the Edited nearest-Neighbor Rule.* volume 6-6, pages 448–452. IEEE Transactions on Systems, Man and Cybernetics, 1976.

[Valls 00] A. Valls. *Development of a method for Multiple Criteria Decision Making based on negation fuctions. Chapter 2: State of the art.* Thesis proposal, Artificial Intelligence Program, UPC, 2000.

[van Rijsbergen 79] C. J. van Rijsbergen. Information retrieval. Butter-worths, London, second edition, 1979.

[Vilà 02] R. Vilà and M. Montaner. *Implementació d'un Sistema Multiagent Distribuit Format per Agents Personals que Recomanen Restaurants Aplicant Raonament Basat en Casos i Tècniques de Trust*, May, 2002. Projecte Fi de Carrera en Enginyeria Informàtica, Universitat de Girona.

[Webb 96] G. Webb and M. Kuzmycz. *Feature Based Modelling: A Methodology for Producing Coherent, Consistent, Dynamically Changing Models of Agents' Competencies.* In User Modelling and User-Adapted Interaction, volume 5, pages 117–150, 1996.

[Widmer 96] G. Widmer and M. Kubat. *Learning in the presence of concept drift and hidden contexts.* In Machine Learning, volume 23, pages 69–101. Kluwer Academic Publishers, 1996.

[Wilson 72] D. L. Wilson. *Asymptotic Properties of Nearest Neighbor Rules Using Edited Data.* In IEEE Transactions on Systems, Man and Cybernetics, volume 2-3, pages 408–421, 1972.

[Wilson 97] D. R. Wilson and T. R. Martinez. *Improved Heterogeneous Distance Functions.* In Journal of Articial Intelligence Research, volume 6:1, pages 1–34, 1997.

[Wilson 00] D. R. Wilson and T. R. Martinez. *Reduction Techniques of Instance-Based Learning Algorithms.* In Machine Learning, volume 38:3, pages 257–286. Kluwer Academic Publishers, 2000.

[Yan 95] T. W. Yan and H. Garcia-Molina. *SIFT – A Tool for Wide-Area Information Dissemination.* In Proceedings of the 1195 USENIX Technical Conference, pages 177–186, 1995.

[Yu 00] B. Yu and M. P. Singh. *A social mechanism of reputation management in electronic communities.* In cooperative information agents, CIA-2000, pages 154–165, 2000.

[Zhang 98] Z. Zhang and Q. Yang. *Towards Lifetime Maintenance Of Case Based Indexes For Continual Case Based Reasoning.* In F. Giunchiglia, editor, Artificial Intelligence: Methodology, Systems and Applications. Springer, Berlin, pages 489–500, 1998.