E.ALKILIÇGİL

USER MODELING IN MOBILE ENVIRONMENT

ERDEM ALKILIÇGİL

DECEMBER 2005

METU 2005

USER MODELING IN MOBILE ENVIRONMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY
ERDEM ALKILIÇGİL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2005

Approval of the Graduate School of METU

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. İsmet ERKMEN Co. Supervisor	Prof Dr. Aydan ERKMEN Supervisor

Examining Committee Members

Prof Dr. Aydan Erkmen (METU, EEE) _____

Prof. Dr. İsmet Erkmen (METU, EEE) _____

Assoc. Prof. Veysi İşler (METU,CENG) _____

Assoc. Prof. Gözde Bozdağ Akar (METU, EEE) _____

Assis. Prof. Afşar Saranlı (METU, EEE) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Erdem Alkılıçgil

Signature :

ABSTRACT

USER MODELING IN MOBILE ENVIRONMENT

Alkılıçgil, Erdem
M.S., Electrical and Electronics Engineering

Supervisor : Prof. Dr. Aydan Erkmen
Co-Supervisor: Prof. Dr. İsmet Erkmen

December 2005, 129 pages

The popularity of e-commerce sites and applications that use recommendations and user modeling is increased recently. The development and contest in tourism calls attention of large-scale IT companies. These companies have started to work on recommendation systems and user modeling on tourism sector. Some of the clustering methodologies, neighboring methods and machine learning algorithms are commenced to use for making predictions about tourist's interests while he/she is traveling around the city. Recommendation ability is the most interesting thing for a tourist guide application. Recommender systems are composed of two main approaches, collaborative and content-based filtering. Collaborative filtering algorithms look for people that have similar interests and properties, while content-based filtering methods pay attention to sole user's interests and properties to make recommendations. Both of the approaches have advantages and disadvantages, for that reason sometimes these two approaches are used together. Chosen method

directly affects the recommendation quality, so advantages and disadvantages of both methods will be examined carefully.

Recommendation of locations or services can be seen as a classification problem. Artificial intelligent systems like neural networks, genetic algorithms, particle swarm optimization algorithms, artificial immune systems are inspired from natural life and can be used as classifier systems. Artificial immune system, inspired from human immune system, has ability to classify huge numbers of different patterns. In this paper ESGuide, a tourist guide application that uses artificial immune system is examined. ESGuide application is a client-server application that helps tourists while they are traveling around the city. ESGuide has two components: Map agent and recommender agent. Map agent helps the tourist while he/she interacts with the city map. Tourist should rate the locations and items while traveling. Due to these ratings and client-server interaction, recommender agent tries to predict user interested places and items. Tourist has a chance to state if he/she likes the recommendation or not. If the tourist does not like the recommendation, new recommendation set is created and presented to the user.

Keywords: User modeling, User Profiling, Recommender Systems, Collaborative Filtering, Evolutionary Algorithm, Artificial Immune Systems, Tourist Guide.

ÖZ

MOBIL ORTAMDA KULLANICI MODELLEME

Alkılıçgil, Erdem
M.S., Electrical and Electronics Engineering

Tez Yöneticisi: Prof. Dr. Aydan Erkmen
Ortak Tez Yönetizisi: Prof. Dr. İsmet Erkmen

Aralık 2005, 129 pages

Kullanıcı modelleme ve tavsiye sistemleri kullanan uygulamaların ve e-şirketlerin popülerliği son günlerde artmaktadır. Turizmde yaşanan gelişmeler ve yarış bilişim şirketlerinin dikkatini çekmektedir. Bu şirketler, turizm sektöründe kullanıcı modelleme ve tavsiye yapan programlar üzerinde çalışmaya başlamışlardır. Bazı kümeleme metodları, komşu methodları ve öğrenen makina algoritmaları, kullanıcı şehri gezerken, kullanıcının ilgilerini tahmin etmekte kullanılmaktadır. Bir turist rehberi uygulaması için en ilginç şey tavsiyedir.

Tavsiye algoritmaları iki farklı yöntemden oluşur: işbirliği ve içerik-temelli filtreleme. İşbirliği filtreleme algoritmaları benzer ilgi ve özellikteki insanlara bakarken, içerik temelli metodlar tavsiye yapmak için tek bir kullanıcının ilgilerine ve özellikleriyle ilgilenmektedir. İki yönteminde avantajları ve dezavantajları vardır, bu sebeple bazen iki yöntem beraber kullanılır. Seçilen metod direct olarak tavsiye kalitesini etkilemektedir, bu sebeple her iki yönteminde avantajlaıı ve dezavantajları dikkatlice incelenmiştir.

Konum ve servis tavsiyesi sınıflandırma problemi olarak görülebilir. Yapay sinir ağları, genetic algoritmalar, küme parçası optimizasyonu algoritmaları ve yapay sinir

algoritmaları gibi biyolojiden etkilenen yapay zeka uygulamaları sınıflandırma algoritması gibi kullanılabilirler. İnsanların bağışıklık sistemlerinden esinlenilen yapay bağışıklık sistemleri çok sayıda farklı şablonu sınıflandırabilme yeteneğine sahiptir. Bu tezde bir turist rehberi uygulaması olan ve yapay bağışıklık sistemini kullanan ESGuide uygulaması incelenmiştir. ESGuide uygulaması turistler gezerken onlara yardım eden bir istemci sunucu uygulamasıdır. ESGuide uygulamasının iki bileşeni vardır: Harita aracı ve tavsiye aracı. Harita aracı kullanıcı haritayı kullanırken kullanıcıya yardım eder. Turistler bölge ve nesneleri gezerken derecelendirmelidir. Bu derecelendirme istemci sunucu etkileşimiyle, tavsiye aracıyla kullanıcının ilgilendiği bölgeleri ve nesneleri tahmin etmeye çalışır. Turistin bu ytavsiyeyi beğenip beğenmediğini belirtme şansı vardır. Eğer turist tavsiyeyi beğenmezse yeni bir tavsiye oluşturulur ve kullanıcıya gösterilir.

Anahtar Kelimeler: Kullanıcı modelleme, kullanıcı profillendirme, tavsiye sistemleri, işbirliği filtreleme, evrime dayalı algoritmalar, yapay bağışıklık sistemleri, turist rehberi

ACKNOWLEDGMENTS

I would like to state my pleasure to Prof. Dr. Aydan Erkmen and İsmet Erkmen for motivating and guiding me through the study of thesis.

Special thanks to members of the examining comitee: Prof.Dr. Veysi İşler, Assoc.Prof. Gözde Bozdağı Akar and Assis.Prof. Afşar Saranlı for their valuable comments.

I also want to express my apprecion to my family for their patience during thesis.

To My Family

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objective and problem specification

Our objective is to develop an intelligent tourist guide application that guides a tourist during his/her trip around a city according to his/her growing interest based on his/her profile as a user. Based on that objective the tourist guide application should provide tourists interactive services in which, users should be able to select a location with the help of an interactive map; So that they can learn the opening and closing times of attractions and their locations. The tourist guide should be able to display the cost of a service or the price of an item. The information that the tourist guide application presents, should be updated dynamically with a learning ability about the growing, changing interests of the user. The tourist guide should be making recommendations to the user not only for locations, but also about services and items. The tourist should be able to evaluate the system through ratings given to places and services after taking the offered recommendations. User interface must be simple, so that the user can select and learn what he/she wants in a few seconds by clicking one or two buttons. Information taken from the user must be easy to fill, that is to say that tourist guide application should not insist the user to fill all the information in a form. The aim of this thesis work is to generate such a recommendation system using a highly versatile, expandable, dynamic and robust method for producing high quality recommendations. User's higher rankings should affect the future recommendations of the guide system to the same user, while being evolutionary, adapting to the changing needs of the mobile user. This specific aim is towards personalizing touristic activities and services based on tailored recommendations during touristic navigation based on demographic data of the user and his/her own ever-growing interests (namely the user profile).

1.2 Motivation

Recently, information retrieval systems and personal services have started to draw attention of large tourism agencies. These agencies provide web pages for supplying services and information to their customers. Some of the agencies allow customers to make their reservation using web pages where user can make a travel plan, reserve hotels, suggest prepacked tour plans etc., and also some of the agencies have started to use travel information systems. Using automated agents that has ability to offer personalized travel plans and services, customer satisfaction which is the most important criterion in leading the competition among companies, can be increased. These agents designate the user needs and interests for a successful travel plan.

The intention of information retrieval systems is to reduce information overload by filtering irrelevant information. The objective of these systems is to acquire useful information while eliminating unnecessary information. Determining needlessness of information is a crucial task for people, since struggling for finding useful information is a time consuming activity. Nowadays people do not want to be bothered with useless information, because time means money for large companies which use intelligent agents in place of human employees. For such an intelligent system the most important thing is to meet user needs in a short time period, and also agreeableness and usefulness of this information have a major importance. Thus, filtering algorithms are needed to find relevant information [1]. Personalization and intelligent recommendation services for tourists which is one of our aims, require user modeling where a user profile is constituted by observing user preferences and interactions. To achieve user modeling, some of the automated agents just use simple filters, while some of them use learning and classification tasks [2]

In the past, since the mobile devices had low computing power, these devices could only be used for giving textual information about the places that the tourist wanted to travel. Some of the more developed ones could support these informations with video and audio for increased visitor satisfaction. Technology for mobile devices improving day after day, mobile environments can use intelligent agents to serve personalized services, tourist guides that learn from user actions, and tailored recommendation to the changing needs and interests of a user.

Feedback from user is an important criterion for perfectioning recommendation, therefore in a recommendation system, user should rate the elements (places,

cuisines, services), so that the recommender agent can understand if the user likes the element or not. The rated elements could be used as a reference for future elements that the user would enjoy to be recommended in the future. The recommender system can be thought of as a black box that learns the user profile from his/her input. The preliminary input to the system could be demographic data about the user. Critical parameters in the user profile like former experiences, life style, age, gender etc. that will affect the objects or services that the user enjoys, should be directly modeled in the system likewise the user interests should also be presented. Then the input to the black box should be demographic data about the user, and user interests. Since the aim is to find appropriate recommendations, the output of the black box will be recommended items. By using inputs gathered from the user, the system should first classify users, and recommend for each different class of users, different types of items with different characteristics (locations, movies, books etc.).

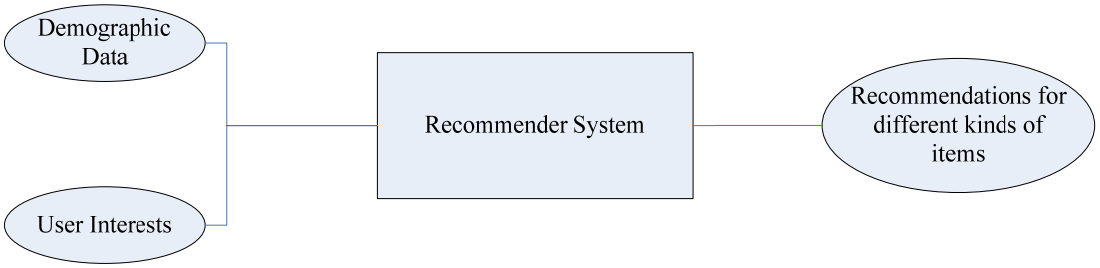

Figure 1.1 System inputs and output

1.3 Goals

Since the screen size of mobile devices is limited, the interfaces of the recommender system should be simplistic. To achieve this goal, selecting items on the screen should be used instead of filling forms. The architecture of the system should allow the user to enter his/her demographic data which should be categorized into features. Features which can help us discriminate between user classes or more specifically user profiles. These features can include age, gender, occupation nationality etc. Since interests of a user directly affects his/her choices, the user interests could be selected from the user interface. After gathering user input, because of security concerns these data should be decoded into byte arrays. In the system, user should be able to rate the services or locations that he/she likes while traveling around the city,

thus, the tourist guide system should have an interactive map, that shows the position of the places that user wants to travel. By clicking on a place on the virtual map, the user should gather information about that place and/or rate it. Virtual map can prevent user from getting bored while using the tourist guide.

Since demographic data of a user affects his/her choices and enjoyment level from different kinds of services, this data is a significant concern while personalizing a service. For example, while a small kid enjoys traveling places like Disneyland, an older man would like old British castles. After getting the user input, the system should then pre-classify these users using their demographic data. Pre-classification of users is needed to reduce the information overload. Next, users should be clustered according to their interests. In this part of the algorithm, one should take into consideration that users can make mistakes while specifying their interests. Therefore, this part of the algorithm should be error tolerant, and should be able to handle such made in specifying ones own interests.

The system should be equipped with a decision making component that can associate the user profile which the user is belonging. Since one of our objects is that the system can adjust itself to new trends, these user profiles should be dynamically, ally updated. This component could be designed as a classifier, so classification algorithms or pattern recognition algorithms are good candidates to be incorporated in the system architecture.

The next part of the recommendation algorithm is the learning component. The system should learn from user interactions, and learned user profiles should reside in a database.

After the system learns the user profiles, it should be able to associate items that will be recommended, to user demographic data. The system can handle this situation with the help of ratings that the user has given to the items, that should determine the level of user enjoyment from these items. In general, gathering information from users in such a tourist guide system is not so easy, sometimes users may not fill in the fields or rate all the services that they have used. The learning component should handle this incompleteness or imprecision situation, and constitute user profiles even when the dataset gathered from users is sparse.

Finally, the system will recommend an item among these by users. While designing this component, it must be noticed that the recommender agent should have the

ability to recommend different kind of items, and the component should be expandable to recommend different kinds of new items also being able to recommend totally new items that have not been previously rated by any user using the application.

1.4 System Flow for a Recommender System

The recommender system in Figure 1.2 should have two pre-classifier components, one of them clustering users by their demographic data and the other classifying them according to their interests. After such a preprocessing of users, the system should filter user profiles which are different than the recent user's profile to model the user. In general such a filtering system falls in one of the two categories: Collaborative filtering and content-based filtering. As a phase subsequent to filtering, the system will learn the profile of the user, adjusting the system variables and saving them into the database. Finally, since the memory of the learning system has to be associative, the user is associated and presented with a set of recommended items. Having overviewed the component sequence in the system architecture let us focus on the methodology that can be used for the different processing of the architecture. We initially introduce the filtering methodologies that can be used.

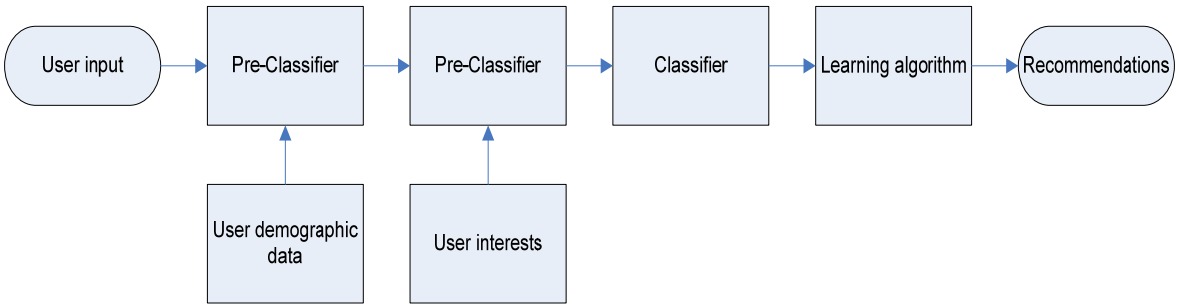

Figure 1.2 System Flow for a Recommender System

1.4.1 Case-Based Recommendations

Content-based filtering pay attention to the content that a single user is interested in. This method uses elements rated "good" and interests of similar user to construct the user profile that has to be used to predict user votes for unobserved content [3]. Content-based methods, or namely search-based recommender systems try to recommend items that have the same category or property, with the help of highly

rated items. For example if the user have rated an horror movie, system will recommend him movies that are directed by the director of this movie, horror books, the movies that the actor/actress of the movie played. This algorithm can recommend different products and services, but recommendation quality will be inadequate [4]. For example, if a user visits www.eskisehirspor.com (A web page that is prepared by supporters of a famous soccer club in Turkey) the system will learn from this action and will make recommendations that include other soccer clubs' web pages, web pages related to Turkey, home pages of Eskişehirspor's supporters, betting web sites about soccer, web sites about sport other than soccer etc. Thus, user actions in this web site directly affects the recommended items. If the user just looks at for example the latest scores of a league match, the algorithm should not recommend him home page of the supporters, but if he/she clicks on the chat button, and speaks with other supporters than the system understands that the user is interested in communicating with other supporters and would recommend also home sites of other supporters.

1.4.2 Collaborative Filtering

Collaborative filtering is used for predicting user behaviors, interests or actions by observing a group of similar users, namely this method is based on multi-user correlations. Similar users are called like-minded users in the literature. Collaborative recommendation systems use user interests or demographic data to predict user ratings and group like-minded users. User decisions are taken into consideration in collaborative filtering instead of knowledge extracted from content. Former used objects or services are taken into account during construction of user profile with such a filter [5]. The former experience of the user, or properties that differentiate user from other user should not be neglected. Lifestyle and interests and former experience of the user generally affect his/her choices in buying a product or choosing a service. Properties including age, gender, occupation, salary etc. separates user from other users, and yields differences in choices. The effect of these attributes on user choice can be modeled with different weights that are adapted dynamically to make high quality recommendations [6].

The input of a collaborative filtering algorithm is usually three vectors: users, objects and user ratings to these objects. The aim is to correlate user ratings and relate users with objects. Ratings is generally single valued (good or bad, like or dislike) but

these types of ratings are found to be insufficient for recommending newer products, and consequently, multi-valued (one-to-five, one-to-ten) rating systems are used [7]. The collaborative filter should equally reflect changes in trends, and adapts itself to changes in the popularity of items. At the same time it has to be error tolerant, handling the errors in user judgments made upon objects of interest [8].

1.4.3 Differences and similarities between collaborative and case-based approach:

The major advantage of using content-based algorithm is privacy, since the system is not concerned with multi-user information and their correlations as it is the case in collaborative filtering. In collaborative filtering privacy could be provided by encoding user demographic data and interests.

Content based algorithms has an advantage of recommending unrated items where in most of the collaborative algorithms this is not possible. Recommending items using content-based algorithm can be explained by the reason behind any of their recommendation, because the system looks for categories of items, and tend to recommend similar categorized items, while collaborative filters can be thought as black boxes based on user correlations. Since collaborative filters use user similarity, explaining the reason for their recommendations is not easy [9].

In content-based recommendation, sometimes defining the cases is not so easy, and errors during categorization of items can cause reduction of recommendation quality, or bad recommendation of wrong items. As an example, if a group of fans of a singer is entered into the system, listening and enjoying a song of that singer the system will then recommend a vast number of songs from that same singer. The quality of the recommendation consequently decreases, because the user doesn't have to like other songs of the same singer [10]. Since the item categories are static, newly added items have to be classified into predefined cases which cause narrow choices of items: if a user likes an item, every time items that is in the same category with that item will be recommended to the user, causing reduced diversity [11]. In collaborative filtering, recommendations do not depend on classification of content, so risk of recommending irrelevant items is eliminated, and diversity do exist [5].

Since collaborative filtering method pays attention to user behaviors instead of content, it can recommend any type of content, such as text, books, music, movies,

web pages etc. making collaborative recommendation response time slower than that for content-based recommendation [6]. Clustering before collaborative filtering is used to reduce computation time. Also dimensionality reduction algorithms are used for elimination of low-frequency items. This approach groups similar customers into clusters with the disadvantage of reducing recommendation quality with respect to low member of classes. If the number of clusters is increased recommendation quality increases but this time computation time also increases. Clusters can be generated dynamically or deterministically [4].

The idea behind the clustering algorithm is dividing the solution space into segments where after clustering similar users are grouped into the same segment. Furthermore, every segment in a solution space corresponds to a recommendation class.

Collaborative filtering has an advantage of constituting user communities [12], meaning that this algorithm make us belong to or differ from other people's experiences, interests and tastes [5]. Since personalization for each sole user is too hard for tourism agencies, a recommendation algorithm used for groups is more useful. In this case using collaborative approach is advantageous over content-based approaches.

Since both of the systems have advantages and disadvantages some, of the systems use a combination of these systems. Classification systems can also be used as collaborative filters. Both of them try to classify the users according to their profiles.

1.4.4 General system flow for a recommender agent

If collaborative filtering approach has to be used as a purposive recommender system, the input to the system would be the recent user's demographic data and his/her interests. A pre-clustering algorithm will then be used to reduce information overload and users in the system will be clustered according to both, their demographic data and their interests, before being input to the collaborative filtering algorithm. In the pre-clustering algorithm also data about user interests and his/her demographic data will be related. User interests will thus affect recommendations, but not determine them directly. If the former user's interests are definitely different than that of the recent user, then this user will be neglected, and thrown out of the system. The algorithm will calculate the closeness of the ratings of the users and update weights in the architecture. In this part of the algorithm, the system can use an

algorithm that will make the output of the system to cover whole solution space. After learning, the system will classify users and try to recommend them a set of items.

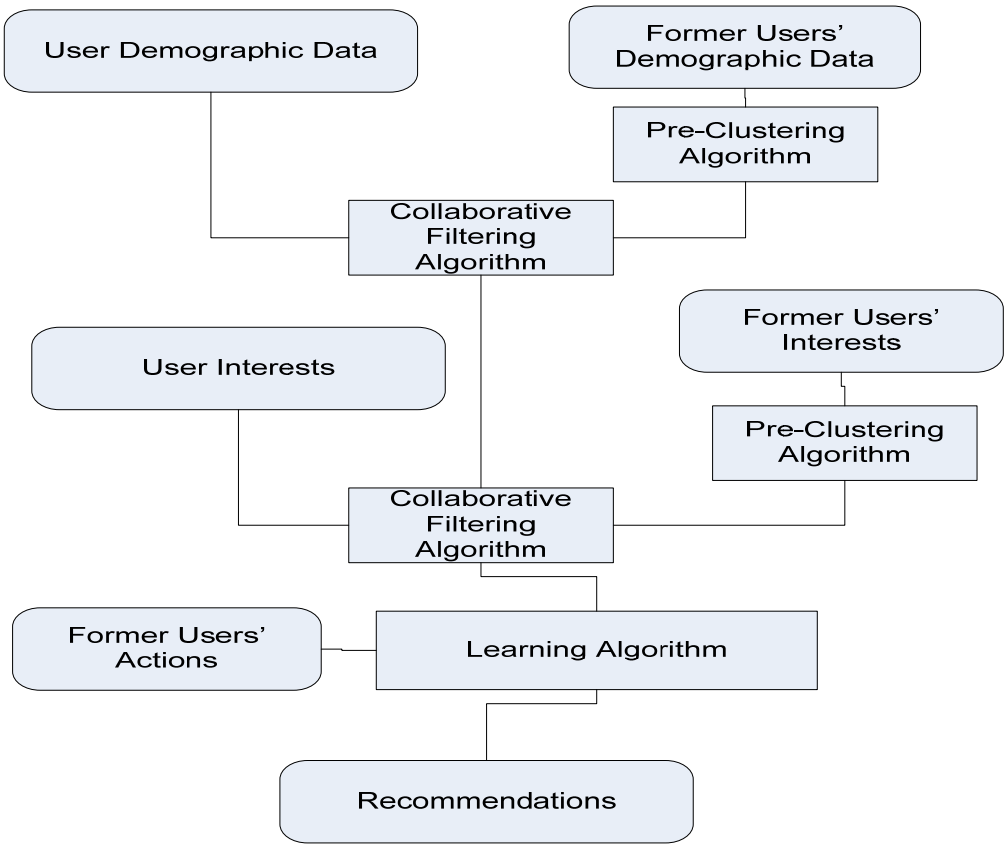

Figure 1.3 Methodology

1.4.4.1 Using genetic algorithms

Genetic algorithms could be used as our learning algorithm. Since generally, genetic algorithms are used to find an optimum solution, it can determine the best fit profiles for a given user. Mutation operator and crossover operators of the genetic algorithm can make the output of the system cover the whole solution space. The unobserved class of users can be classified by the system to generate recommendations to these users. Using genetic algorithms has a disadvantage. The purpose of our system being to recommend different kinds of items or services to the user, for each different type of item that will be recommended to the user, a new genetic algorithm model should

be used, therefore, increasing architectural and processing complexity of the recommendation system. Pre-clustering algorithms are another unknown issue for such a framework. Different kinds of algorithm that have clustering abilities (i.e. K-Means Clustering) should then be used. In such a recommendation system, a genetic algorithm should be used for each of the two collaborative filtering algorithms. Moreover, relating user interests and demographic data still remains as a problem in a GA based recommendation system. Another problem in this structure is that when a user reenters the system, the system will recommend him or her the same items, because, every time genetic algorithms will try to find the best fit recommended items, and the system will recommend every time the same items or services to the user since they are the optimal ones and remain to be so. This situation can be boring for a tourist that will use the system regularly. These disadvantages make the use of genetic algorithms in our system undesirable.

1.4.4.2 Using probabilistic methods

In our architecture, we could use prediction algorithms as the learning algorithm. The user could be clustered by any clustering algorithm based on their demographic data, after that, their interests would be used to eliminate some of these user clusters. After clustering the user, a collaborative filtering algorithm would then be used, so that the user clusters could be put into a prediction algorithm to find his/her future actions.
The system can be designed for predicting user's actions, or selections. In such a system, it is logical to use prediction algorithms like Bayesian Networks. Bayesian Networks learns from a user or a class of users' actions or selections to predict user's future actions or selections. In our proposed system, the objective is to recommend items to the user by profiling this user with the help of similar users that have previously used the system. Our aim is to recommend instead of predicting the user's next movements and actions. For predicting different kinds of actions, such that for example what kind of movie the user will watch, what kind of books he will read etc., different Bayesian algorithms must be used for each different type of actions. If our aim was to predict one type of action than using Bayesian Network would have been efficient, but for many different kinds of actions, implementing a series of different Bayesian Networks makes the recommendation system more complicated.

Moreover, our aim is to avoid combinatory algorithms, and use one method that can pre-cluster and recommend users.

1.4.4.3 Using neural networks

Neural networks are biologically inspired algorithms that have classifying abilities. The algorithm is constructed with differently weighted interconnections of neurons. By adjusting these weights, the neural network classify the input and associates it with an appropriate output. For clustering the users based on their demographic data a neural network could be used but relating user interests with these clusters of users is a problem. A simple weighting algorithm could be used for relating user demographic data and interests. If data gathered from the user is sparse, the output of the neural network could not cover whole solution space since the system is not evolutionary. Another disadvantage of this algorithm exists if similar users enter the system periodically. The system will memorize the output, and every time it will give the same output. Like in Bayesian Networks, for each different kinds of actions, or items a series of neural networks should be implemented which will increase the complexity of the architecture.

1.4.4.4 Using Artificial Immune Systems

Artificial immune systems are evolutionary learning algorithms, taking ideas from human immune system. Like in human immune system, the algorithm tries to classify foreign objects entering the system. When a user enters into the system and fills his/her demographic data and user interests, an antigen object which is an artificial immune cell is created. The artificial immune system uses TCells, BCells and antibodies to classify this user. These objects correspond to the phenomenon taken from natural immune system: TCell corresponds to the demographic data of user, where antibody represents user interests and BCells represents the rated items. Former user's demographic data is symbolized by the TCell population in immune system. The clustering of TCell population and calculation of fitness measure correspond respectively to pre-clustering algorithm and collaborative filtering algorithm in Figure 1.3. Similarly, secretion of antibodies stands for clustering users according to their interests in the architecture. Mutation, positive and negative

selection operators are used as the learning part of the structure. Finally, affinity maturation described in section 3.3.1, represents the recommendation part. The whole desired architecture of Figure 1.3 can be modeled by a single immune system leading to structural simplicity. Also if different kinds of items are wanted to be added, then just adding different kinds of BCell objects is enough, which makes the system extendable. Since the system uses mutation operator, it can recommend unrated elements, and can thus cover the whole solution space.

1.4.5 Contribution

In order to reduce complexity, a single method is desired to handle the whole system flow shown in Figure 1.3. This single methodology is taken from immunology. Our algorithm uses immune cells to model user behaviors, and recommend him/her services and places, such that a collaborative recommendation system is constructed by using an artificial immune system algorithm. This algorithm uses ideas taken from genetic algorithm and human immune system. The proposed algorithm simulates the behaviors of immune system and does not contain any other non-immune phenomenon like Active Recognition Balls which is used to determine neighborhood of immune cells in AIRS algorithm [28]. On the other hand, algorithms such that aiNet [15], SWAMI Framework [26], and AIS [17] ignores TCells and BCells and just simulates behaviors of antibodies and antigens. Our approach models every processes in the recommendation system using every concept in an immune system including antigens, BCell, TCell and antibodies, primary and secondary responses, apoptosis, mutation, affinity maturation, positive and negative selection.

Since the artificial immune system classifies users inherently, there is no need for clustering them before putting them into the immune system algorithm: In the immune system TCell object constructs the user clusters. In most recommendation algorithms either demographic data or user interests are used separately. In our algorithm we use both demographic data and user interests to construct user models. User interests are modeled as antibody objects which gives us a chance to reduce failures that can be caused by false determination of user interests or any other user specific errors. These errors caused by the fact that the user can not determine his/her interests precisely and exhaustively are then repaired by the immune system. This

provides our algorithm an advantage since other collaborative filtering algorithms do not take at all interests into account and thus interest errors cannot even be modeled.

Besides other collaborative filtering algorithms such as “PTV”, recommending only movies [6], “RACOFI” (recommending only music) [8], “Entrée” recommending restaurants [20], “LIBRA” recommending books[9], our algorithm can make recommendations about books, movies, cuisines and locations spanning all kinds of interests. Our algorithm has the additive advantage of high scalability and recommending different types of items. The framework is highly adaptable for use towards other pattern recognition and classification tasks.

In [29], the stereotype concept is used to model a user. To model the user dynamically a pre-clustering algorithm is used, firstly a neural network is used to cluster the user data. After that four neural networks are used to generate stereotypes dynamically. Our algorithm simplifies such a complexity by using a single structure for all the processes in the recommendation flow.

Generally, recommendations should reflect changes in popularity of items, and former user’s interests as well as demographic data effects the recommendations. By adjusting the life counts of BCells and TCells, using positive and negative selection algorithms the system can adapt to changes in trends.

In our immune system algorithm, after the recommendations are presented to the user, he/she can accept or reject the recommendation set. This causes life counts of BCell objects to be changed, directly effecting future recommendations. This novel idea is not considered in any other immune system or collaborative filtering algorithm.

Different than other collaborative filtering approaches, user demographic data is not saved into the database in our architecture: The privacy is as much high as in case-based algorithms. The disadvantage of collaborative filtering algorithm is that if user inputs to the system are sparse, recommendation quality decreases. This disadvantage does not exist in our methodology of immune system modeling. Applying mutation and two selection algorithms handle the sparsity in user data, and still, high quality recommendations are obtained.

1.6 Outline of the Thesis:

In the first chapter an introduction is given followed by sections on objective, methodology and contribution. In the second chapter an exhaustive survey is presented. The 3rd chapter introduces the proposed method in its analytical details. In chapter 4 the implementation software of our architecture is presented. Chapter 5 dwells with experiments and performance analysis. And finally, chapter 6 concludes the thesis work and discusses positive future works.

CHAPTER 2

RELATED WORK

2.1 Recommendation Systems

In the literature the recommender systems can be classified as collaborative and case-based recommender systems. In systems such as PTV [19], CASPER [11], Electronic Cultural Agency [13], Entrée [23], ITR [2], LIBRA [9] case based algorithms is used, and in Particle Swarm Optimization [6], Group Mark [1] collaborative filtering is used.

The works on immune systems based on the ideas gathered from the genetic algorithms [24]. After that some primitive algorithms use antibodies and antigens to classify the user (i.e. aiNet [15], SWAMI [26] and EachMovie [22]). Then algorithms tend to use the immune network algorithms such as CLONALG [25]. Finally algorithms such that AIS [17] and AIRS [28] have used a component named active recognition balls.

PTV - A Personalised TV Listings Service **[19]**

Nowadays channel numbers are increased because of new cable and satellite services. Audiences have difficulties in selecting appropriate programs from an increasing number of 200 channels and 4000 programmes per day. Digital TV vendors use Electronic Program Guide (EPG) which is an on-line TV listing which can be shown on the screen. However this solution is not a feasible for 200 channels and more. The aim of PTV as a client server system is to filter out irrelevant programme and provide a personalized EPG for each individual. The server side is composed of schedule database, profile database, programme case-base database and recommender component. Explanations about all TV listings reside in the schedule

database. Profiles of each registered user are stored in profile database. While, programme content descriptions such as programme title, genre information, the creator and director etc. are stored in the programme case-base database. The recommender component is the intelligent part of the system which job is to select appropriate programmes for user. This component uses a hybrid recommendation engine that combines content-based and collaborative recomendation techniques. In PTV profiles are constructed two ways, users can manually select viewing preferences or recommender system updates new programme preferences. For PTV, content based recommendation is done by recommending programmes that are similar to the programmes in a user's positive programme list and dissimilar to those in the negative programme list. A set of features is originated from each case in programme case-base, similarities between those being defined as the weighted-sum of the similarity between corresponding case features. Profile schemas that are constructed from each raw user profile corresponds to a content summary of the programme preferences contained in a user profile.

The similarity between a profile and a given programme case can then be computed using the standard weighted-sum similarity metric:

$$PrgSim(Schema(u),p) = \sum w_i \bullet sim\left(f_i^{Schema(u)}, f_i^p\right)$$

where $f_i^{Schema(u)}$ and f_i^p are the i_{th} features of the schema and the programme case respectively.

A ranked list of programme recommendations is produced where the similarity between a given programme and the target profile scheme corresponds to the estimated quality of the recommendation; programmes that are very similar to the target profile are judged to be good recommendations. The best r content-based recommendations are selected as candidates for the user's personalised guide, wehere r is a predefined number in the system.

In collaborative filtering, instead of similarly rated programmes, users that have similar ratings are selected where similarity is computed based on Pearson corellation. Pearson correlation can be a good candidate as a similarity measure for our recommender system. PTV can use the case-based filtering, since it recommends only the TV programmes; but in our case since we recommend different kinds of items, defining the cases is not easy.

***CASPER (Case-Based Profiling for Electronic Recruitment) project* [11]**

The aim of this work is to built an intelligent search engine for the internet. CASPER is tested on Jobfinder web site(www.jobefinder.ie). CASPER PCR is a personalised job case retrieval system. It uses a client-server based system. Similarity-based retrieval component resides in the server side and case-based personalisation component resides in the client side of the system. When a new search query is entered, the server-side similarity-based search engine selects similar job cases. Subsequently, in the client part a personalization engine filters irrelevant jobs based on the user profile. Thus, remaining jobs are relevant to user interests. In the first phase, the job search engine simply takes the user query and compares it with the jobs in the job-case base to find best matches. This comparison is not applied in a tradional way; instead of finding exact matches, jobs that have higher correalation scores are listed. Job cases are defined as a set of fixed features like job type, salary, key skills, minimum experience etc.

Different similarity metrics are defined for different case features. In the server-side of the application implicit needs of user is ignored. In client-side users can apply for jobs, email themselves the description of relevant jobs etc. The personalization in the client-side is seen as a classification task and a nearest-neighbor type classification is used in the system. Graded job cases(liked or disliked / good or bad) in a target user's profile is used as training data. Each applicant retrieved job is correlated with each job base to find k-nearest jobs where k must be set experimentally. To recommend jobs, k neighborhoods are selected by a relevance score. The relevance of a candidate job j to user profile P is the sum of the product of the similarity values

of the k nearest neighbours to j by their class value.
Since server side information can be updated easily, in our system we can save the information on a server like it is done in CASPER, despite saving on these information on clients.

***Electronic Cultural Agency (Agencia Cultural Eletrónica, ACE)* [13]**

This system is a personal recommendation service to recommend leisure activities in Barcelona, Spain. The system is adapted to an electronic newspaper to make recommendations about people interests and tastes.

The aim in this application is to make recommendations, with having a little information about the content which is like in our case. When the user enters in the system, he/she is registered by entering username and password. The users can optionally fill fields about their interests. If the user does not fill these fields, the system has to learn user interests implicitly. Our system should start with a welcome screen, after that user data should be filled to construct a user vector.

Informant agent is the component that works with database to make recommendations. Feedback spy agent observes user actions to personalize the system. Suggestion spy agent controls the virtual community of users' suggestions. Collection agent collects information form different sources such as web pages or public databases. After Central agency collects information from interface and collector agents, it filters and retrieves documents to the user.

As it is designed in this system, in our application we should use different agents in different kinds of works, i.e map agent could be used to handle the user interactions on the virtual map. When collector agency sends documents to Document Modeling Agent, document model is generated from tye content of document. Dealer Agent is stimulated to inform the arrival of new item to recommend, and filters uninteresting items and writes these items to database. Informant agent will display tese items to the user. User modelling agent tries to learn more from the user. The document model is formed by Document Model Agent. The document model consists of a vector-space model including a list of word and weight pairs. Words are breaked into their roots. Interest of a user u for a document d, $p_t(u,d)$ is defined as:

$$p_t(u,d) = k \cdot p_c(u,d) + (1-k) \cdot p_s(u,d)$$

Content interest $p_c(u,d)$ is defined as the interest of user u for the content of document d and $p_c(u,d)$ is defined as interest of user u for document d given the votes made from other users to document d.

Document model and user model is used for estimation of content interest for user u and document d.

$$p_c(u,d) = sim(u',d') = \sum_{i=1}^{n} (u'_i\, d'_i)$$

The estimation of the social interest for user *u* and document *d* is given by:

$$p_s(u,d) = \overline{v}_u + k\sum_{i=1}^{n} w(u,i)(v_{i,d} - \overline{v}_i)$$

$\overline{v}_i$ is the main vote of user i, $v_{i,j}$ is the vote of user i to document j. W(u,j) is the similarity factor between user i and user j.

A user model is contructed for each user via user votes on documents or suggestions made by user. Votes can be given in two ways: explictly or implicitly. In explicit voting user votes the document as positive or negative. In implicit voting, if a user consumes more time on a document a popup appears asking if this page is interesting to the user. If a user writes suggestions to a document, this is evaluated as a good vote. The space vector model is used to construct user model.

Content-based learning is used to learn a basic user model. A vector that contains the best group of words is originated to generate user model. Using the formulas below;

$$E(W,S) = I(S) - \left[P(W = present)I(S_{w=present}) + P(W = \neg present)I(S_{w=\neg present})\right]$$

and

$$I(S) = \sum_{c\in\{+,-\}} - p(S_c)\log_2(p(S_c))$$

documents are classified as positively voted and negatively voted.

The algorithm is defined as:

IF the new word was included in the old model **THEN** the new *tf*, *idf* and *information*

gain are averaged with the new values obtained.

IF the new word was not included in the old model but there is space enough to include
that word **THEN** the new word is included with the *tf* ,*idf* and its *information gain*.
IF the new word was not included in the old model and there is not space enough **THEN**
we replace the new word with another word that has a *tf-idf*Information Gain* higher than that word

Secondly, social filtering(collaborative filtering) algorithm is used. A space vector model is used where each vector is a user with similar interest. Similarity measure is defined as;

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

a is the actual user and i is one of the users in the system. $V_{a,j}$ is the vote of user a to document j. If the similarity measure of a user reaches a threshold value than user a is included in social vector of i. After testing the system, we get accurate recommendations for users.

RACOFI: A Rule-Applying Collaborative Filtering System [8]

RACOFI is an online music recommendation system lets users to rate modern Canadian music in five dimensions of impression, lyrics, music, originality and production. The idea in this system is sharing items that users view as valuable. The system collects ratings from each user and calculates a similarity measure between user ratings and recommends items to each of the user. If most of the users rate in the same way, this means that they will be happy to recommend other good rated items as well. Since collaborative filtering is content-neutral, RACOFI could be easily configured as a movie, book or tourist recommender. Objective and subjective meta-data is defined in the RACOFI system. Objective data consists of bibliographical(the date of item, the author's name, the name of the album etc.) information of the content; where subjective meta-data depends on the user ratings. After logging user

can give ratings to objects, view information about objects or add new objects.ratings are given zero-to-ten scale. More ratings that a user gives to items, more accurate recommendations could be made. Properties have weights, this means some categories could be defined more importance than others. For example, if a user enjoys from one singer's album than he/she could enjoy other albums of the same singer.

For a recommendation tool, a client application that can get ratings for objects that are stored in the database could be used. Client should rate the objects, and highly rated items could be recommended to other users that use this client. In fact, this application could be determined as a recommender system, but this system could not satisfy the user. Because, system will recommend the same items to different users. Only semi-learning algorithm will change the ratings of objects according to their popularity, and recommend them from highest rated item to lowest rated one. In our algorithm we should decide also the user interests and relate it to the demographic data of the user, which is absent in RACOFI.

Algorithms:

To find a correlation between feature vectors of a user, mostly used algorithm is Pearson correlation algorithm. More faster but less accurate algorithm is Bias Mean algorithm. Some applications force users to rate a set of items before using the program. Some users are giving rarely worse ratings; this problem is handled by normalization of rating vectors.

Amazon.com Recommendations: Item-to-Item collaborative filtering [4]

Amazon.com uses personalization algorithms to recommend their products. Interface of the site changes for each different customer according to their interests, such as while baby toys are presented a new mother programming titles are shown to a software engineer.

Item-to-item collaborative filtering merges three different algorithms: traditional collaborative filtering, cluster models and search-based methods. In most algorithms, a feature vector is constructed from user ratings. To compare these feature vectors usually a similarity measure is used.

Calculating cosine between two vectors is a common approach:

$$similarity(\vec{A}, \vec{B}) = \cos(\vec{A}, \vec{B}) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|}$$

Clustering before collaborative filtering is used to reduce computation time. Also dimensionality reduction algorithms are used for elimination of low-frequency items. Using clustering is important for improving the evoluation interval of the system, since we are planning to design a tourist guide application, since the visitors visiting the city can be in huge numbers as millions, we should use an effective pre-clustering algorithm.

Content-based methods, or namely search-based recommends items that have the same category or porperty. With the help of highly rated items, the system try to find similar categorized items and give recommendations. For example if the user have rated an horror movie, system recommends him movies that are directed by the director of this movie, horror books, the movies that the actor/actress of the movie played. This algorithm will recommend different products and services, but recommendation quality will be inadequate.

In amazon.com by clicking your recommendations link users can rate the products, write suggestions, and filter their recommendations. Amazon.com uses an algorithm called item-to-item recommendation algortihm. This algorithm uses similar rated items instead of searching for similar users. A set of items that customers desire to buy together is attempted to be constructed.

```
For each item in product catalog, I1
  For each customer C who purchased I1
    For each item I2 purchased by
       customer C
      Record that a customer purchased I1
       and I2
  For each item I2
    Compute the similarity between I1 and I2
```

In this method cosine measure is used as a similarity measure. The algorithm finds the similar purchases of each user and constructs a items matrix. From this matrix, application forms a list of popular or corralated items and recommends them to user.

The Restaurant Guide Entree: **[23]**

The Entree application uses multi-scaled semantic ratings. For example a user shows interests to a restaurant, the system recommends restaurants with similar properties. In our approach we use user interests instead of the items properties, since defining these properties for different kinds of items is very difficult. Think that user finds this restaurant expensive, and presses the button "Less $$". System eliminates the restaurants expensive than firstly selected one, and creates a list of restaurants cheaper but similar properties with first one. User can filter the properties of restaurants represented in recommendation list. A user can take one of each eight actions, that are There are eight possible actions a user can take: "Less $$," "Nicer," "More Creative," "More Traditional," "Quieter," "Livelier," "Change Cuisine," and "Browse". One of the ten ratings can be given to a restaurant, these ratings including "Entry Point", "Exit Point" and eight different actions. If a user ranks a restaurant more than once, only last ranking is taken into consideration and former ones will be canceled. A projection of multi-dimensional approach onto one dimensional recommendation approach could be used. For example, if the user wants a cheaper restaurant this means a negative rating, and if user likes the restaurant tihs means a positive ranking. Another approach is observing exactly veritable preferences of users. List containing these restaurants is very sparse, but large numbers of users is needed for this kind of approach. In final approach, recommendations are made depending on the similarity of user ratings. For example, user A selects "Nicer" and user B selects "More Traditional" even though these two user ranks restaurant negatively, both users are thought as dissimilar users. A table is constructed by applying these rules; a) A rating is maximally similar to itself. b) "Browse" is not similar to any other rating. c) Some ratings have natural opposites: "Livelier" / "Quieter", "Traditional" / "Creative". "Entry Point" and "Exit Point" are behaved as positive rankings. The cosine similarity metrics is used as a similarity measure.

Particle Swarm Optimization Recommender System **[6]**

Particle swarm optimization algorithm is simply a profile matching algorithm, and arranges recommendations with preferences of users. Particle swarm algorithm is an

evolutionary algorithm; each particle in this algorithm has a position, velocity and acceleration component. While the swarm moves through the solution space, it gets samples of positions from its route.

Each sample position have a fitness value, in our case this value corresponds to like hood of user to an object. In this work, the particle swarm optimization algorithm is used as a collaborative recommendation approach. 22 features from MovieLens database is used in the system. These features contain movie rating, age, gender, occupation and 18 movie genre (action, adventure, animation, children, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, sci-fi, thriller, war and western). Since MovieLEns does not contain in formation about interests of user, it can not be used for our recommendation system. User profile is constructed via these 22 features. Except rating feature, other 21 features are filled once in the beginning of the application. After constructing the profile, a neighborhood algorithm consists of three different duties such as Profile Selection, Profile Matching and Best Profile Collection. Profile selection is defined as choosing a predefined number of user profiles from user profile database. Euclidean distance is used as a profile matching algorithm.

$$euclidean(A,j) = \sqrt{\sum_{i=1}^{z}\sum_{f=1}^{22} w_f * diff_{i,f}(A,j)^2}$$

where: A is the active user
j is a user provided by the profile selection process, where $j \neq A$
z is the number of common movies that users A and j have rated.
w_f, is the active user's weight for feature f
i is a common movie item, where *profile(A,i)* and *profile(j,i)* exists.
$diff_{i,f}(A,j)$ is the difference in profile value for feature f between users A and j on movie item i.

The particles dynamics are calculated by using the formula below;

$$vi = wvi + c1r1(xpbest,i - xi) + c2r2(xgbest - xi)$$
$$if(|vi| > v\max) \quad vi = (v\max/|vi|)vi$$
$$xi = xi + vi$$

x_i is the current position of particle i

x_{pbest} is the best position attained by particle i

x_{gbest} is the swarm's global best position

v_i is the velocity of particle i

w is a random inertia weight between 0.5 and 1

c_1 and c_2 are spring constants whose values are set to 1.494

r_1 and r_2 are random numbers between 0 and 1

A prediction formula is used to find a recommendation set.

$$predict\_vote(A,i) = mean_A + k \sum_{j=1}^{n} euclidean(A,j)(vote(j,i) - mean_j)$$

k is a normalising factor such that the sum of the
euclidean distances is equal to 1.
vote(j,i) is the actual vote that user j has given on item i
n is the size of the neighbourhood.

Since our aim is to recommend items, instead of finding their probability of selection, this algorithm can not be used for our propose.

GroupMark, a prototype based on SELECT architecture **[1]**

In GroupMark application bookmark-based approach is used. Each user is associated with a user group that have similarly or partially bookmarked the web pages. In GroupMark application group owner adds new users to a group by looking their relevance to this group. Imagine a scenerio that a user A creates a bookmark list A_b, and a recommender group G_A is created with the help of this bookmark list and bookmark lists of other users. Then

$$G_A = \{X: |A_b \cap X_b| \geq t\}$$

Where t is the similarity threshold, bookmarks of other users is denoted by X_b. The set of recommendations is defined as

$$R_A = \{X_b: X \in G_A\}$$

Exploiting Re-ranking Information in a Case-Based Personal Travel Assistant [10]

Personal Travel Assistant is a case-based recommendation system that assists users in arranging flights. The system learns user preferences automatically and recommends travel preferences. Recommendations are constructed by examining user feedback. After users request a flight plan, intelligent system will return an optimal recommended flight plans list, and helps user to choose one of the optimal solutions. The system learns from previous preferences of travel plans of user, and when the user request a travel plan from the system by using a simple web page, the system produces a list of travel plans. PTA system uses Case-Based Reasoning algorithm to make these recommendations. For example, if a user selects a list of offers from set S;

$$S = \{O_1, \ldots, O_n\}.$$

where

$$O_i > \{O_1, \ldots, O_{i-1}, O_{i+1}, \ldots, O_n\}$$

After that a similarity measure between offers is defined;

$$O_1 > O_2 \text{ if } Sim(O_1, O_i) > Sim(O_2, O_i)$$

Ratings given to the offers are calculated with the help of similary measures,

$$Rating_i = 10 \times \frac{Sim_i - Sim_{lowest}}{Sim_{highest} - Sim_{lowest}}$$

n number of features containing the origin, destination, travel time, distance, number of hops, price etc is used in travel offer case. These features are weighted by thier importance. A similarity measure is used that combines similarity measures (at feature level) and feature weights (depended to importance of features).

After making a recommendation list for similarity scores, the list is presented to user. If the user selects not an optimal solution, than the system tunes the feature weights.

***ITR: a Case-Based Travel Advisory System* [2]**

Intelligent Travel Recommender (ITR) system allows user to create a travel plan, select locations, attractions and leisure activities. ITR plans not only the locations

that user should travel but also it gives advices for attractions, leisure activities etc. ITR uses case-based reasoning to make recommendations. Case-bases are travel wishes, travel bag, user, and reward (case outcome). A travel bag consists of (locations, accommodations, attractions, activities). Interactive query management is used to handle user interaction with the system. Firstly, in ITR system case similarity is calculated by calculating similarity between items, then a collaborative approach is applied, thus a correlation between other session is measured. When a user wants to make a travel plan, ITR system asks him a few questions to learn his wishes about the travel plan. Questions include "With who will you travel?", "Transportation vehicle", "When are you leaving", "Where do you come from?" etc. After that user add's locations and activities to his/her travel bag. The user also select some filters for travel plan. After that the recommendations are presented to the user.

Case-Base is constructed from four parts: travel wishes and constraints TWC, travel bag TB, user U and reward R. Solution space can be shown as

$$CB = TWC \times TB \times U \times R$$

Some of the features that forms Travel wishes and constraints part are Travel Party (alone, with partners, a few friends, etc.); Budget (a constraint on the day allowance); Transportation mean (car, camper, bike, etc.); Accommodation Type (hotel, apartment, etc.); Departure (country of residence); Time (the period and the duration of the travel) etc.

Travel bag contains a set of items that user collects before recommendation process.

The user is modeled with a set of feature vectors such that

$$U = \prod_{i=1}^{n} U_i$$

The rank is the rate of a case given by the user. To make recommendations we have two sets of items.

$$X \supset S, R.\ S = \{x_1, \ldots, x_m\}$$

Where S is the set of items that user collected and R is the set of items that former users have collected that have similar context with the ones in user's list.

After user makes a query a list of locations is presented to the user including the top ten cases. Finally this set of locations are ranked according to the similarty of cases that they possess. Euclidean Overlap metric is used as similarity measure.

$$heom(x,y) = \frac{1}{\sqrt{\sum_{i=1}^{n} w_i}} \sqrt{\sum_{i=1}^{n} w_i d_i(x_i, y_i)^2}$$

where:

$$d_i(x_i, y_i) = \begin{cases} 1 & \text{if } x_i \text{ or } y_i \text{ are unknown} \\ overlap(x_i, y_i) & \text{if the } i\text{−th feature is symbolic} \\ \frac{|x_i - y_i|}{range_i} & \text{if the } i\text{−th feature is finite integer or real} \end{cases}$$

range i is the difference between the maximum and minimum value of a numeric feature, and overlap(x_i, y_i) = 1 if x_i is not equal to y_i and 0 otherwise. The weights 0≤ wi≤1, i = 1,.,.,.,n are assigned dynamically according to the query definition.

LIBRA (Learning Intelligent Book RecommendingAgent) **[9]**

Database of the system is constructed with the help of pages taken from amazon.com. A one-to-ten scale rating system is used for rankings. Textual meta-data is used as content information. Bayesian Learning system is used for learning user profiles. In the system data about each title is extracted by using pattern-based information-extraction system. title, authors, synopses, published reviews, customer comments, related authors, related titles, and subject terms are trated as additional content about a book. For a book to have enough content information it should have at least one abstract, review or customer comment. When user enters the system, he/she selects and rates a set of training books with one-to-ten ranking scale. Bayesian text classifier for applying learning task in the system. Our aim is not to predict the rating of each item exactly, in fact the aim is to sort total ranking of book titles to recommend. Then the books are classified as positevely rated or negatively rated. Books that have ratings predicted from 0 to 5 is described as negatively rated, and books have ratings 6 to10 are named as pesitively rated.

To calculate posterior probability of each class given a document D, equation below is used;

$$P(c_j|D) = \frac{P(c_j)}{P(D)} \prod_{i=1}^{|D|} P(a_i|c_j)$$

$P(c_j)$ is the probability of each class.

$P(w_k|c_j)$ is the probability of word or token, $w_k \in V$.

a_i is the i_{th} word in the document.

|D| is the length of the document in words.

Posterior category probability of a book is determined by the equation below;

$$P(c_j|B) = \frac{P(c_j)}{P(B)} \prod_{m=1}^{S} \prod_{i=1}^{|d_m|} P(a_{mi}|c_j, s_m)$$

where

d_m is a vector of documents one for each slot.

s_m denotes the m_{th} slot.

$P(w_k|c_j, s_m)$ is the probability of each word given the category and the slot.

S is defined as the number of slots.

a_{mi} is the i_{th} word in the m_{th} slot

and if we neglect some of the parameter calculation equations, finally

A value which is named as Strength to predict the value of a book to be classified as positive or negative.

$$Strength(w_k, s_j) = log(P(w_k|c_1, s_j)/P(w_k|c_0, s_j))$$

After the profile of the user is learned, positive categorized books are used to predict ranking of other books, top positevely rated books are shown to the user.

2.2 Artificial Immune Systems:

Using Genetic Algorithms: **[24]**

Genetic Algorithms are evolutionary algorithm, constructed by James Hook. Genetic Algorithms (GA) is based on genes that use mutation and crossover to maintain diversity. In GA, the solution space is covered with a gene population, and the aim is to evolve an optimized population member which is the optimal solution. Our propose is to find a consensus solution instead of finding an optimum solution. This means our aim is to find the path, music, facility etc. that everyone with a similar profile likes, we are not aiming to find the most chosen path, music, facility etc. Therefore genetic algorithms is not appropriate method for our application. In fact, GA is a good starting point, because GA has basic concepts for an evolutionary

algorithm. The GA without crossover is a reasonable model for clonal selection. Our algorithm must discover a set of pattern matching antibodies that match a set of antigen patterns; also it must maintain diversity. 64 bit strings are used for antibodies.

For each antigen, in the population there is one antibody string that matches some substring contained within the antigen. Antigen and antibodies are said to be matched if their bit strings are complementary. The degree of match is quantified by a matching function M: Antigen x Antibody → R. GA has disadvantages in multiple peak problems, because of its strong convergence tendencies. On multiple peak problems, genetic drift will lead the GA to (randomly) converge on one of the peaks. Common pattern's (schemas) are used with the motivation of the immune system to recognize bacteria partially on the basis of existence of unusual molecules [24]. Firstly set first half of the bits to 1 and remaining randomly, next vice versa. Select p of antigens and find a match score for each antibody, take the average of match scores for each antibody.

We can subdivide the antigen population

```
11******  }
**11****  }quarter-length schemas
****11**  }
******11
```

Maximum fitness=L + S / 2

L= Length of strings (8 above)

S= Length of schemas (2 above)

Diversity algorithm:

- Antigen is chosen at random.
- A sample of antibody population of size σ is chosen randomly without replacement.
- Antibody matched with highest match score added its fitness. The fitness of all other antibodies remains unchanged.
- This process is repeated for many antigens (typically three times the number of antibodies).

Fitness Algorithm:

Chose a sample of ρ antigens randomly from set of antigens (with replacement) Total antigen population remains fixed throughout a run of GA.

Compute match score

The fitness of antibody is the average match score computed over the sample of ρ antigens.

Hamming distance does not affect the discovery of maintenance of peaks.

Fitness Sharing Algorithm:

It causes population members to cluster around peaks in a search space by reducing a population member's fitness to account for the proximity of other individuals in the population.

$$f'_{\mathrm{i}} = \frac{f\mathrm{i}}{\sum_{j=1}^{N} Sh(d\mathrm{ij})}$$

f'_{i} = Shared fitness

$f\mathrm{i}$ = Fitness of individual

$d\mathrm{ij}$ = Distance between i, j

$Sh(d\mathrm{ij})$ = Sharing function

$$Sh(d\mathrm{ij}) = \begin{cases} 1 \text{ if } d\mathrm{ij}=0 \\ 1 - \frac{\mathrm{dij}}{\sigma\mathrm{s}} \text{ if } d\mathrm{ij}<\sigma_s \\ 0 \text{ Otherwise} \end{cases}$$

σs dictates a cutoff distance, which no sharing will occur.

aiNET Algorithm: **[15]**

In aiNet model the antibodies are connected each other with links that have associated connection strengths. The connection strength between antibodies defines the similarity between two antibodies. aiNet is an immune network model. Different than other immune network models, aiNet algorithm is discrete not continuous.

- **Ab**: available antibody repertoire (**Ab** $\in S_{N\times L}$, **Ab** = **Ab**$_{\{d\}} \cup$ **Ab**$_{\{m\}}$);
- **Ab**$_{\{m\}}$: total memory antibody repertoire (**Ab**$_{\{m\}} \in S_{m\times L}$, $m \le N$);
- **Ab**$_{\{d\}}$: d new antibodies to be inserted in **Ab** (**Ab**$_{\{d\}} \in S_{d\times L}$);
- **Ag**: population of antigens (**Ag** $\in S_{M\times L}$);

- $\boldsymbol{f}_i$: vector containing the affinity of all the antibodies $\boldsymbol{Ab}_i$ (i = 1,...N) with relation to antigen $\boldsymbol{Ag}_j$. The affinity is inversely proportional to the Ag-Ab distance;
- **S**: similarity matrix between each pair $\boldsymbol{Ab}_i$-$\boldsymbol{Ab}_j$, with elements $s_{i,j}$ (i,j = 1,...,N);
- **C**: population of N_c clones generated from **Ab** (**C** $\in {}_{L\,N_c}S_{\times}$);
- **C***: population **C** after the affinity maturation process;
- $\boldsymbol{d}_j$: vector containing the affinity between every element from the set **C*** with $\boldsymbol{Ag}_j$;
- ζ: percentage of the mature antibodies to be selected;
- $\mathbf{M}_j$: memory clone for antigen $\boldsymbol{Ag}_j$ (remaining from the process of clonal suppression);
- $\mathbf{M}_j^*$ resultant clonal memory for antigen $\boldsymbol{Ag}_j$;
- σ_d: natural death threshold; and
- σ_s: suppression threshold.
- N is the total amount of antibodies in Ab
- round(×) is the operator that rounds the value in parenthesis towards its closest integer.
- $D_{i,j}$ is the distance between the selected antibody i and the given antigen Ag_j.

AiNet algorithm is constructed as;

1. At each iteration, do:

1.1. For each antigenic pattern $\boldsymbol{Ag}_j$, j = 1,...,M, ($\boldsymbol{Ag}_j \in$ **Ag**), do:

1.1.1. Determine its affinity $f_{i,j}$, i = 1,...,N, to all $\boldsymbol{Ab}_i$. $f_{i,j} = 1/D_{i,j}$, i = 1,...,N:

$$D_{i,j} = \| \text{Ab}_i - \text{Ag}_j \|, \; i=1,...,N$$

1.1.2. A subset $\mathbf{Ab}_{\{n\}}$ composed of the n highest affinity antibodies is selected;

1.1.3. The n selected antibodies are going to proliferate (clone) proportionally to their antigenic affinity $f_{i,j}$, generating a set **C** of clones: the higher the affinity, the larger the clone size for each of the n selected antibodies;

1.1.4. The set **C** is submitted to a directed affinity maturation process (guided mutation) generating a mutated set **C***, where each antibody k from **C*** will suffer a mutation with a rate α_k inversely proportional to the antigenic affinity $f_{i,j}$ of its parent antibody: the higher the affinity, the smaller the mutation rate:

$$\boldsymbol{C}_k^* = \boldsymbol{C}_k + \alpha_k (\boldsymbol{Ag}_j - \boldsymbol{C}_k); \; \alpha_k \propto 1/f_{i,j}; \; k = 1,..., N_c; \; i = 1,..., N.$$

1.1.5. Determine the affinity $d_{k,j} = 1/D_{k,j}$ among $\boldsymbol{Ag}_j$ and all the elements of **C***:

$$D_{k,j} = \| \mathbf{C}_k^* - \boldsymbol{Ag}_j \| \;\; k=1,...,N_c$$

1.1.6. From **C***, re-select ζ% of the antibodies with highest $d_{k,j}$ and put them into a matrix $\mathbf{M}_j$ of clonal memory;

1.1.7. *Apoptosis*: eliminate all the memory clones from $\mathbf{M}_j$ whose affinity $D_{k,j} > \sigma_d$:

1.1.8. Determine the affinity $s_{i,k}$ among the memory clones:

$$s_{i,k} = \| \mathbf{M}_{j,i} - \mathbf{M}_{j,k} \|, \; \forall i,k$$

1.1.9. Clonal suppression: eliminate those memory clones whose si,k < ss:

1.1.10. Concatenate the total antibody memory matrix with the resultant clonal memory M_j^* for Ag_j: Ab{m} ← [Ab{m};M_j^*];

1.2. Determine the affinity among all the memory antibodies from Ab{m}:

$$s_{i,k} = \| \boldsymbol{Ab}_{\{m\}}^i - \boldsymbol{Ab}_{\{m\}}^k \|, \;\; \forall i,k$$

1.3. Network suppression: eliminate all the antibodies such that $s_{i,k} < \sigma$ s:

1.4. Build the total antibody matrix Ab ← [$Ab_{\{m\}}$;$Ab_{\{d\}}$]

2. Test the stopping criterion.

To determine the total clone size Nc generated for each of the M antigens, the following equation was employed:

$$N_c = \sum_{i=1}^{n} round\left(N - D_{i,j}.N\right)$$

This algorithm could be used as a framework for our algorithm, since the system ignores some of the immune cells such as BCells and TCells, our algorithm should expand this algorithm.

Each Movie: **[22]**

The main difference between genetic algorithms and immune systems is that immune system is not designed for optimization. In immune system the aim is not finding a single optimum. In fact, we intend to identify a subset of good matches on which recommendations can be based. Building a neighborhood of user profiles is the main idea of each movie. The preferences of user, usually a set of votes on an item, comprise a user profile, and these profiles are compared to build a neighborhood. User profile is constructed with a string of numbers; each number presents the "vote" for an item. After encoding, next thing to decide is to calculate similarity measure. The most common method is correlation based like Pearson or Spearman. Vector based (e.g. cosine of the angle between vectors), probabilistic methods and k-nearest neighborhood can also be used. Users do not rate all the movies in the list, this cause a problem known as "curse of dimensionality". Single value decomposition is used to handle this problem. It both improves efficiency and increase overlap.

In the first instance we intend to build a model where known user preferences are our pool of antibodies and new preferences to be matched is the antigen in question, if the concentrations of those antibodies that provide a better match are allowed to increase over time, we should end up with a subset of good matches. In fact we are not interested to find best match. We require a set of antibodies that are close match but which at some time distinct from each other for successful recommendation.

Each movie data links a person with a movie and assigns score (taken from the set {0, 0.2, 0.4, 0.6, 0.8, and 1.0} User demographic information (e.g. age and gender) is provided but not used. Content information about movies (e.g. category) is not used.

Algorithm:

Similarity Measure (Pearson is used):

$$r = \frac{\sum_{i=1}^{n}(u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^{n}(u_i - \bar{u})^2 \sum_{i=1}^{n}(v_i - \bar{v})^2}}$$

u,v are users.

n = movies for which both u and v voted.

u_i = vote for user u for movie i.

- Initialize AIS
- Encode user for whom to make predictions as Antigen Ag.
- WHILE (AIS is not stabilized) & (Reviewers available) DO
- Add next user as antibody Ab
- Calculate matching scores between Ab and other antibodies.
- WHILE (AIS at full size) & (AIS not stable) DO
- Iterate AIS
- OD
- OD

At each iteration an antibody's concentration increased by an amount which depends on its matching to the antigen and decreased which depends on it's matching to other antibodies. Antibody concentration will slowly decrease over time. Antibodies with sufficiently low concentration are removed from the system.

Stabilization means that a sufficient number of "good" neighbors have been identified and therefore a prediction can be made. "Poor" neighbors would be expected to drop out of AIS.

Earlier antibodies have advantage to recently ones because recent ones had no chance to increase in concentration. To remove this disadvantage:

- Reset AIS (set all antibodies to initial concentrations)
- WHILE (no antibody at max. concentration) DO
- Iterate AIS

- OD

The Pearson Measure can be used as a similarity in our system, since it works with the ratings gathered from users.

Immune Network based on CLONALG algorithm: **[25]**

CLONALG algorithm is based on clonal selection algorithm:

1. Randomly initialize a population of individuals (M);
2. For each pattern of P, present it to the population M and determine its affinity (match) with each element of the population M;
3. Select n_1 of the best highest affinity elements of M and generate copies of these individuals proportionally to their affinity with the antigen. The higher the affinity, the higher the number of copies, and vice-versa;
4. Mutate all these copies with a rate proportional to their affinity with the input pattern: the higher the affinity, the smaller the mutation rate, and vice-versa.
5. Add these mutated individuals to the population M and re-select n_2 of these maturated (optimized) individuals to be kept as memories of the system;
6. Repeat Steps 2 to 5 until a certain criterion is met, such as a minimum pattern recognition or classification error.

Due to immune network theory immune cells carry receptors named as idiotopes, the antibodies that can recognize the same antigen constructs a network. If an immune cell recognizes a self cell then network suppression takes place, and if an immune cell recognizes an antigen, then network activation and cell proliferation is occurred.

Immune network algorithm could be driven with the help of CLONALG algorithm as:

Given a set of patterns (P) to be recognized, the basic algorithm runs as follows:

1. Randomly initialize the network population;
2. For each antigenic pattern in P apply the CLONALG algorithm that will return a set of memory cells (M*) and their co-ordinates for the current antigen;
3. Determine the affinity (degree of matching) among all the individuals of M*;
4. Eliminate all but one of the individuals in M* whose affinities are greater than a given threshold. The purpose of this process is to eliminate redundancy in the network by suppressing self-recognizing elements;

5. Concatenate the remaining individuals of the previous step with the remaining individuals found for each antigenic pattern presented. This will result in a large population of memory individuals M;

6. Determine the affinity of the whole population M and suppress all but one of the self-recognizing elements. This will result in a reduced final population of memory cells that recognize and follow the spatial distribution of the antigens.
7. Repeat Steps 2 to 6 until a pre-defined stopping criterion is met, such as a minimum pattern recognition or classification error.
This algorithm is not suitable for our system because our aim is to use all of the immune system cells like TCells, BCells and antibodies.

***Artificial Immune System Algorithm Based on SWAMI framework*: [26]**

The SWAMI (Shared Wisdom through the Amalgamation of Many Interpretations) is a collaborative filtering framework. In SWAMI user vector is formed as:
User = {{id_1, score1}, {id_2, score2},…, {id_n, $score_n$}} where id is the movie unique id, and score is the user rating given to the movie. Pearson measure is used as a similarity measure .
Initialise AIS
Encode user for whom to make predictions as antigen Ag
WHILE (AIS not stabilised) & (Reviewers available) DO
Add next user as an antibody Ab
Calculate matching scores between Ab and Ag
Calculate matching scores between Ab and other antibodies
WHILE (AIS at full size) & (AIS not stable) DO
Iterate AIS
OD
OD

The antibody concentration increases when the antigen - antibody matching increases and antibody concentration decreases if the antibody - antibody matching increases. AIS iteration will be managed by the equation:

$$\frac{dx_i}{dt} = c\left[\begin{pmatrix}\text{antibodies}\\ \text{recognised}\end{pmatrix} - \begin{pmatrix}\text{I am}\\ \text{recognised}\end{pmatrix} + \begin{pmatrix}\text{antigens}\\ \text{recognised}\end{pmatrix}\right] - \begin{pmatrix}\text{death}\\ \text{rate}\end{pmatrix}$$

$$= c\left[\sum_{j=1}^{N} m_{ji}x_i x_j - k_1\sum_{j=1}^{N} m_{ij}x_i x_j + \sum_{j=1}^{n} m_{ji}x_i y_j\right] - k_2 x_i \qquad (1)$$

Where:

N is the number of antibodies and n is the number of antigens.

x_i (or x_i) is the concentration of antibody i (or j)

y_i is the concentration of antigen j

c is a rate constant

k_1 is a suppressive effect and k_2 is the death rate

m_{ji} is the matching function between antibody i and antibody (or antigen) j

A simplified version of the above equation;

$$\frac{dx_i}{dt} = k_1 m_i x_i y - \frac{k_2}{n}\sum_{j=1}^{n} m_{ij}x_i x_j - k_3 x_i$$

Where:

k_1 is stimulation, k_2 suppression and k_3 death rate

m_i is the correlation between antibody i and the (sole) antigen

x_i (or x_i) is the concentration of antibody i (or j)

y is the concentration of the (sole) antigen

m_{ij} is the correlation between antibodies i and j

n is the number of antibodies.

Since the former antibodies are saturated in concentration and newer antibodies have no chance to be proliferated, the algorithm should be iterated as follows;

Reset AIS (set all antibodies to initial concentrations)
WHILE (No antibody at maximum concentration) DO
    Iterate AIS
OD

Current AIS models are not so scalable because they require huge datasets for B-Cell objects [27]. These models require storage and manipulation of very large numbers of B-Cells.

AIS Algorithm: **[17]**

AIS is an artificial immune network model based on antibodies [1]. It has content addressable memory, recognition and self organizing ability. A bone morrow object is used in AIS to generate B-Cells; also it decides where the antigen will be put in the immune network, controls the B-Cell population. B-Cells in the network are responsible for creating antibodies. Stimulation level determines the strength of antibody antigen binding and affinity between different B-Cells. Data is encoded as binary strings. The disadvantage of this system is needs the storage and manipulation of immune cells for each user. Instead of this method, we can use some of the immune cells for pre-clustering the users.

AIRS Algorithm**: [28]**

Artificial immune recognition system is a supervised learning algorithm that simulates ideas learned from immune systems. Affinity measure in AIRS is defined as the closeness of antigen and antibody objects. Closeness is calculated by Euclidean distance between antigen and antibody vectors. B-Cells are named as Active Recognition Balls (ARBs) that consist of an antibody and ARBs have a stimulation threshold.

In AIRS consist of four main stages: Algorithm starts with normalization and initialization. Then memory cells are determined and ARBs are generated. After that ARBs compete for development of candidate memory cells. Finally, winner candidate cells become established memory cells.

In AIRS, the following notation is used;

- MC = A set of memory cells.
- mc = An individual memory cell.
- ag.c = The class constructed by Antigen ag, where $ag \in C = \{1, 2, ..., nc\}$ *and nc* is the number of classes in the data set.
- *mc.c* = the class of a given memory cell, *mc*, where $mc.c \in C = \{1, 2, ..., nc\}$

$$MC_c \subseteq MC = \{MC_1 \cup MC_2 \cup \ldots MC_{nc}\}$$

- ag.f and mc.f = the feature vector of a given antigen and memory cell, ag and mc, respectively.

- AB = the set of ARBs, or the population of existing cells.
- MU = a set of mutated clones of ARBs.
- *ab* = a single ARB where $ab \in AB$.
- *ab.c* = the class of a given ARB, *ab*, where $ab.c \in C = \{1, 2, ..., nc\}$

$$AB_c \subseteq AB = \{AB_1 \cup AB_2 \cup \ldots AB_{nc}\}$$

- *ab.stim* = the stimulation level of the ARB *ab*.
- *ab.resources* = the number of resources held by the ARB *ab*.
- *TotalNumResources* = the total number of system resources.

In normalization phase, all possible reactions in the immune cell level is guaranteed to be in the range of [0,1]. After normalization phase affinity threshold is calculated.

$$\text{affinity threshold} = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \text{affinity}(ag_i, ag_j)}{\frac{n(n-1)}{2}}$$

n is the number of training data items (antigens) in question, ag_i and ag_j are the i_{th} and j_{th} training antigen in the antigen training vector, and affinity(x,y) returns the Euclidean distance between the two antigens' feature vectors. Then ARB population is seeded randomly from the memory cells. After that ARBs that match with the antigen is determined,

$$mc_{match} = \text{arg}max_{mc \in MC_{ag.c}} stimulation(ag, mc)$$

mc_{match} is the memory cell matched by the antigen ag. Stimulation(x,y) is defined as;

$$stimulation(x, y) = 1 - \text{affinity}(x, y)$$

After mc_{match} is determined, these memory cells are used for generating new ARBs, this is done by mutating memory cells that are matched to the antigen.

After mutation, the algorithm tries to produce candidate memory cells for classifying the antigen. By presenting the antigen to the ARB population stimulation level of the ARB is determined. If the ARB population exceeds the allowed number of resources, then least stimulated resources will be removed from the population.

If the candidate memory cell is stimulated by the antigen more than the memory cell, it is replaced with the memory cell. This is the last step in training process. The classification in AIRS is performed by a k-nearest neighborhood. K-most stimulated

memory cells are used for used for data classification. Since this algorithm uses ARBs which is not defined in the human immune system, the algorithm can not be used as our artificial immune recommender system. Stimulation threshold phenomenon could be used in the classification stage of our algorithm where the ratings of the user are not filled yet, and the artificial immune system should be stimulated by a function.

CHAPTER 3

PROPOSED METHOD

3.1 Human Immune System

The immune system is a defense system of the human body. The system is composed of cells, molecules and organs that can detect pathogens. Pathogens (or antigens) are the organisms that damage the human body. The human immune systems detect these organisms and classify them as nonself. Immune system is a remarkable parallel and distributed adaptive system. It uses learning, memory and associative retrieval to solve recognition and classification tasks [20].

The function of the human immune system (HIS) can be seen as classification of proteins in the body into two classes: self, nonself and potentially harmful. [20] The classification ability of the human immune system is important for people who studies machine learning. The immune system has no central controller. If an antigen enters the body only that part of the immune system is activated.

The immune system is composed of innate and adaptive immune system [16]. Adaptive immune systems are made of lymphocytes. Lymphocytes have the ability to recognize antigens. Lymphocytes have two different types: B cells and T cells: T-cells can be divided into two subgroups, cytotoxic T cells and helper T cells. Type I and type II major histocompatibility complex (MHC) proteins play a major role in foreign cell recognition [16]. The B cells develop in bone marrow and T cells develop in the thymus [4] The T cell regulates the activity of immune responses by producing a variety of factors. T cell controls the creation of B cells [18]. T cells are trained by antigen processing cells; they ingest foreign proteins, break them down and, as a final step, attach them to MHC molecules prior to displaying the antigen MHC complex on the cell's surfaces [18].

B cells carry Immunoglobin (IgM and IgD) on their surfaces. All immunoglobins on a given cell are identical and capable of binding antigen [18]. B cells secrete antibody molecules. These molecules are Y shaped receptor molecules bound on the surface of a B cell with the primary role of recognizing and binding, through complementary match, with an antigen [16]. The parathope is the variable region of the antibody that matches with the antigen's epitope. It is variable because it changes for a perfect match with antigen. The strength and specificity of the Ag-Ab interaction is measured by affinity of their match [16]. T cells also work with B cells to help them divide, differentiate and make antibody. Each B cells manufactures one type of antibody. An antibody produced by a B cell is specific for an epitope and not the entire antigen molecule [4].

If B cells with an affinity (matching score) are sufficient in concentration then stimulation threshold is reached B cell ingests pathogen (phogocytosis) [16]. MHC type II molecules are made after (phogocytosis) When a B cell encounters an antigen, an immune response is elicited, which causes the antibody to the antigen(if they match) so that the antigen can be neutralized If the antibody matches the antigen sufficiently well, its B cell becomes stimulated and can produce mutated clones [2] . Stimulation level depends on the affinity between B cells and antigens.

After stimulation of B cells, these cells undergo two stage selection processes. This process is called as clonal selection. First stage is called as positive selection. A positive response would result in cell proliferation, activation and antibody secretion [16]. In positive selection helper T cell that has a binding to MHC type II proteins are not destroyed [22]. Positive selection acts on the entire maturing cell population, and only cells that demonstrate functional surface receptors capable of recognizing major histocompatibility (MHC) molecules continue maturing, the remaining cells undergo apoptosis (programmable cell death) [18]. After positive selection, another selection process is applied to the cells; this process is called negative selection. A negative response would lead to tolerance and suppression [16]. In negative selection T cells react against self proteins are destroyed [16]. This process protects body from "self destruction" namely cancer. B cell and T cell that recognize the same antigen must undergo direct physical contact with each other. Once this occurs, B cell and T cell divide interact and produce progeny, referred to as clones, which express surface receptors identical to those of their parents [18]. When the affinity threshold of a

specific B cell is exceed by either an exact of approximate match, the specific response is elicited from memory (clone, release antibodies, destroy) [4]. This response seems to be proportional to the degree of affinity. B cells undergo an additional transformation called "affinity maturation" [18]. Those cells that recognize antigens grow in concentration and affinity (affinity maturation), while those that do not die out, a phenomenon usually addressed as the maturation of the immune response (immune learning mechanism). This process of pattern recognition and selection is known as clonal selection [16]. Affinity maturation describes which progeny if activated cells produce antibodies (via point mutations) with a higher affinity cell present on the parent cell [18]. B cell activation and the subsequent creation of high affinity clones via affinity maturation received a good deal of attention from artificial immune system designers [18]. Some of the antibodies retains in the immunological memory, so that any subsequent exposure to a similar antigen leads to rapid immune response (secondary response) [20]. The human immune system exhibits two types of response; The primary response is the learning phase when the immune system learns about patterns in input teaching data, the secondary response represents a pattern recognition process during which the immune system attempts to classify new data relative to data it has seen before.

3.2 Properties of immune systems

Ability to perform classification/pattern recognition: The human immune system can separate a vast number of foreign molecules as self or non-self. This attribute gave an idea to engineers to produce an artificial immune system that can classify objects. The information retrieval systems, especially collaborative filtering algorithms could be seen as a classification task. In a tourist guide application each new user can be seen as a non-self molecule. After recognition and classification of a new visitor (or non-self molecule from immune system point of view), artificial immune system can recommend places or services to visitor.

Ability to generalize from input data: The immune system can generalize the input data since it is evolutionary. This is important for the case that user input is sparse.

Online learning: The artificial immune system can learn and recognize at the same time. If a new user enters the system, the immune system starts to learn from user, and at the same time classification process continues. Since the learning phase and

classification phase is separated from each other, these two tasks could be applied simultaneously.

Learning a huge number of patterns: The immune system can learn a vast number of patterns. Despite the foreign molecules and antigens are too complex and a lot of different molecules are observed the immune system can recognize them, and if they are presented to the immune system again, learning is done in a short period than the first one. This is because the immune system has a memory, and created molecules in the primary response is matured, and presented to the antigen in the second response period.

Associative memory: The most important property of the memory of an immune system is its associative memory. As the models Konerra's sparse distributed memory, Abus's cerebellar arithmetic computer and Marr's theory of cerebellar cortex the immune system is also associative. Cross-reactive secondary response or original antigenic sin is called to associative behavior of the immune system by immunologists [12].

Diversity: The immune system uses mutation and colonial selection to maintain diversity. In artificial immune systems different types of mutation operators are used to maintain diversity. Diversity is an important property for a collaborative filtering system. In most cases returning an action from a user is not an easy task. Users do not want to rate items, or fill demographic data. Collaboration handles that by using somatic mutations. The system fills the sparse data coming from user with appropriate data.

Evolutionary approach: Immune system uses evolutions to make improvements in response time of the system, and in recognizing different types of non-self molecules.

Self regularity: The molecules of the immune system are regulated through the positive and negative selection process, mutation and affinity maturation. Self regulation is supplied by life count of the immune cells.

Distributed system: The immune system has no central controller. The cells and molecules in the system work independently. For example, if a pathogen enters the human body from human's hand, and then only this part of the human immune system is stimulated.

Noise tolerant: In some recommendation systems, since the system dynamically changes over time noise in user data will reduce the recommendation quality. In fact errors caused by the user data can be diminished by the immune recommender system. By the help of two selection algorithm and maturated cells in the system, these errors will be eliminated from the output of the system.

Multi-layered: The system is multi-layered like other biologically inspired algorithm such as, neural networks. The layers process T-Cells, B-Cells and antibodies.

Robustness: If a part of the immune system is lost, then remaining cells can reproduce more powerful immune system.

Scalability: Immune systems could be adapted to other machine learning algorithms. The algorithm can be easily modified, and some part of the algorithm could be changed.

Application domain: Immune systems are used in various fields containing security, pattern recognition, network intrusion detection, anomaly detection optimization, document classification etc.

3.3 Our Artificial Immune System Algorithm

Our artificial immune system is used as a collaborative filtering technique. Due to artificial immune system algorithm used in our application; visitors that came in the city previously, is seen as self molecules in the immune system. Self molecules are previously recognized molecules which have already been professedly entered the system. The immune system classifies these molecules very fast. When a new user comes into this city, he/she is considered as a non-self molecule, in other words antigen. Antigens in the natural immune system are defined as non-self molecules entering the body. Our aim is to classify these visitors, with the help of their profiles and the self molecules, namely the former visitors' profiles. The algorithm is composed of two sub algorithms: Classification and learning algorithm.

3.3.1 Classification of Antigens

Classification phase of our artificial immune system corresponds to the secondary response in the human immune system. When a user enters into the system an antigen object is created. The immune system treats this antigen as non-self and tries

to classify this antigen. Our immune system algorithm uses lymphocytes which correspond to the immune cells in the natural immune system. Lymphocites have two types, one of them is T-Cells and other is B-Cells. Every new visitor produces a T-Cell, and these cells construct a T-Cell population shown in figure 3.1. T-Cell object is the feature vector related to user's demographic data, consisting of age, sex, nationality, education, occupation feature vectors. T-Cell ingests antigen object (new visitor), filling the visitor's encoded data. T-Cell population is constructed randomly from the ImmuneDB database, for correlating the previous visitors with the new visitor and calculating a fitness score. The fitness score is calculated by applying an operator that sums each similar bit pairs that belongs to former and new visitors' encoded demographic data, for each similar bit pair fitness measure is increased by 1.

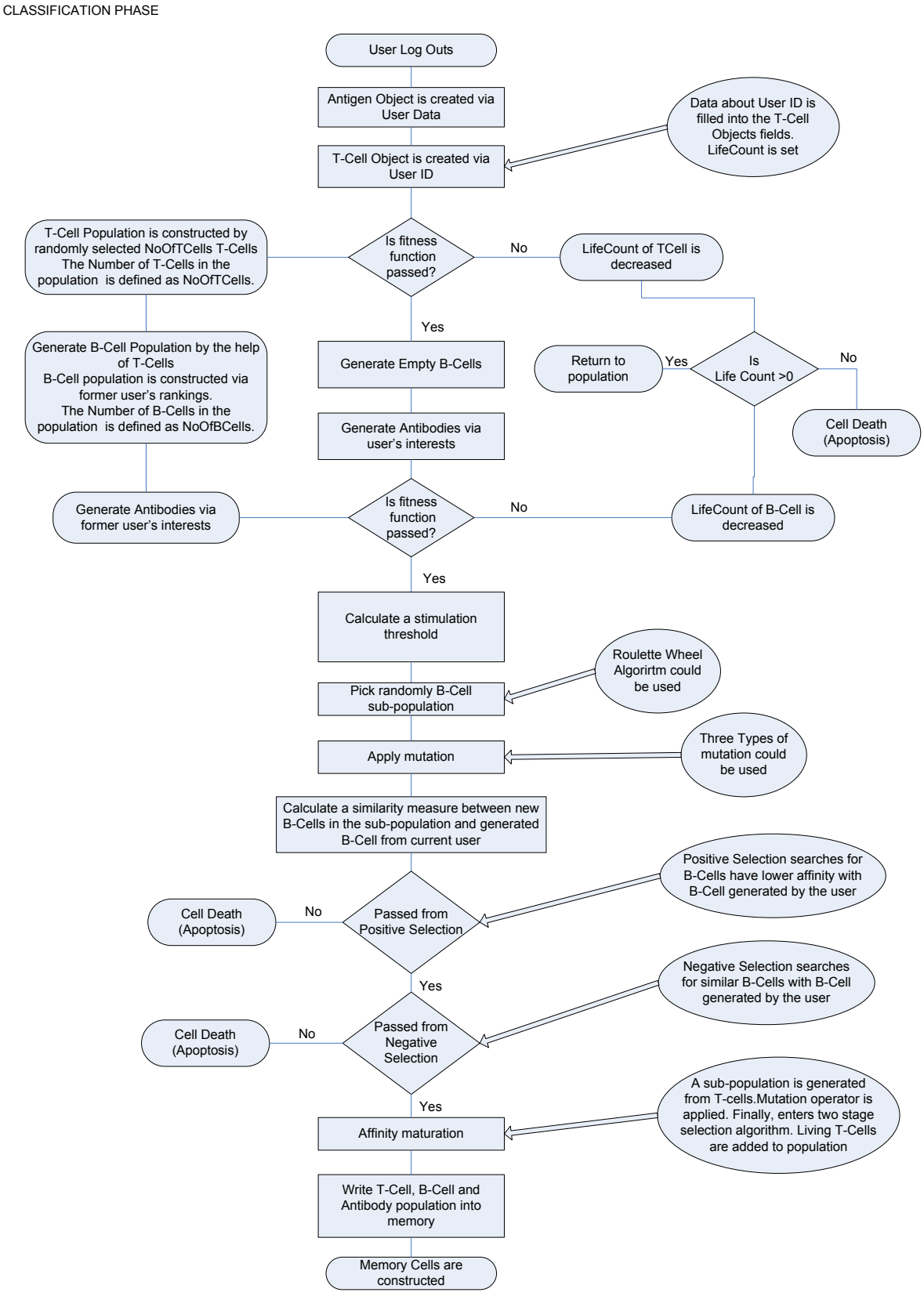

Figure 3.1 Classification Algorithm

The affinity score is calculated as follows;

Current T-Cell: 111011000101110101

T-Cell from T-Cell population: 000100100110110110

The same bits are shown as 1: 000000011100111100

Fitness score: 0+0+0+0+0+0+0+1+1+1+0+0+1+1+1+1+0+0 =7/18=0.38

Equation 3.1

For each TCell in the TCell population an affinity score should be calculated.
Affinity: Affinity is measured between antibodies constructed by new user and those constructed by former visitors. Affinity measure is calculated as the sum of XOR operation between two bits at the same location on each of them. This sum is divided to total number of bits and an affinity score is determined [11].

Antibody of current user: 101011100101110001

Antibody of former user: 011100101110110010

XOR : 110111001011000011

Affinity Score:1+1+0+1+1+1+0+0+1+0+1+1+0+0+0+0+1+1 =10/18=0.56

Equation 3.2

This affinity score is used to determine how new user and former users are similar. T-Cells are compared bit by bit, and similar bits are counted in each feature (in Equation 3.1). Each feature vector's affinity score is compared with a certain threshold value, if it passes this value T-Cell returns to population, if its fitness value is smaller, then it is deleted from the population and life count of the T-Cell which determines how many iterations this cell can be used in the system is decreased by one so that the new lifecount of this cell equals to the lifecount of the same cell minus one. If life count of the T-Cell equals to zero, it is deleted from the database.
After that some of the T-Cells died, and population size is decreased. Remaining T-Cells generates BCell objects which has a TCellID field which determines the TCell that have created this BCell, filled with the ID of its creator TCell , B-Cells are produced with the data belonging to the visitor. Since the new user has not rated for

any item, the fields belonging to these rated items in BCell object is empty. Each T-Cell creates a constant number of BCells. Different types of B-Cells are created for different types of content such that placesBcells are created for locations, moviesBcells are created for movies, bookBcells are created for books etc. B-Cells are content specific. Each TCell in the TCell population constructs a constant number of BCells, so a BCell population is created. Then, antibodies are secreted by each BCell in this population. Antibodies correspond to user interests which are movies, books, cuisine, sports, music, interested historical period of interests etc. After secretion of antibodies, mutation is applied to antibodies. Antibodies secreted by the B-Cells bind to antibody part of the antigen with a complementary match as shown in Equation 3.2. Mutations help us to cover the solution space. Five types of mutation are used:

(1) Multipoint mutation: Random numbers of bits are changed. If the bit value is equal to one, its value is changed to zero. If the bit value is equal to zero bit's value is changed to one.

Antibody: 101011100101110001

Mutated Antibody: 101001100011110101

Random number: 4

(2) Substring generation mutation: From a random start bit to an end bit, a substring is created from antibody string. Bits in this substring are regenerated randomly.

Antibody: 101100100101110001

Mutated Antibody: 101001100011110001

Start Bit: 4

End Bit: 11

(3) Substring conversion mutation: From a random start bit to an end bit, a substring is created from antibody string. If the bit in the substring is zero it is changed to one, else if it is one it is changed to zero.

Antibody: 101100100101110001

Mutated Antibody: 100011011010010001

Start Bit: 3

End Bit: 13

(4) Substring reversion mutation: From a random start bit to an end bit, a substring is created from antibody string. The substring is reverted.

Antibody: 101100100101110001

Mutated Antibody: 101101001001110001

Start Bit: 3

End Bit: 13

(5) Substring copy mutation: From a random start bit to an end bit, a substring is created from antibody string. The substring is copied from a random starting bit.

Antibody: 101100100101110001

Mutated Antibody: 101001011100110001

Start Bit: 6

End Bit: 16

Copy Location: 2

Also, affinity between mutated antibodies is calculated as in Equation 3.2. If this affinity measure can not exceed a threshold, the T cell object that secretes this antibody will perish from the population, and life count of the B cell is decreased. If life count of B cell becomes zero then B cell is deleted from the database. After secretion of antibodies, a stimulation threshold is calculated for each BCells in the BCell population using LifeCount of TCell that the BCell belongs.

$$\text{Stimulation} = \left(\frac{\text{LCT}}{AveLCT}\right)^2 \times TCellFitness^2 \times \text{AntibodyAffinity} \times \text{BCellLifeCount}$$

Equation 3.3

LCT = LifeCount of TCell.

AveLCT = Average LifeCount of TCell population.

After calculating the threshold of each B cell, a subpopulation of B cells is selected by roulette wheel algorithm. In roulette wheel algorithm chance of the selection depends on the similarity measure of B cell. If fitness of a B cell is greater than all others, then it has the highest chance for selection [23].

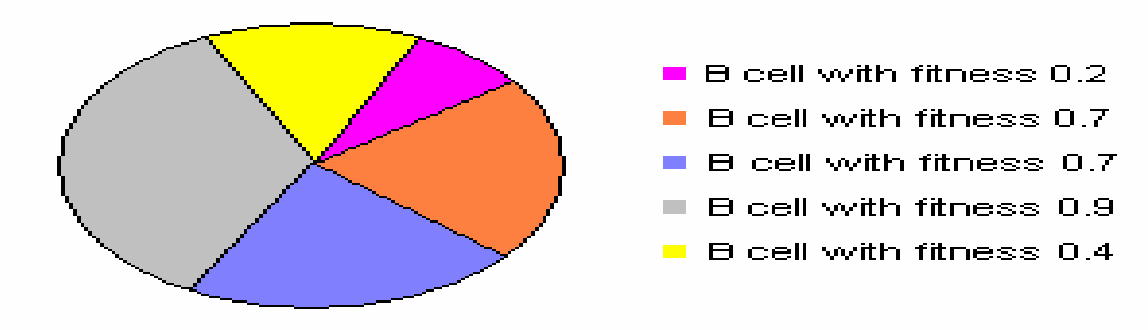

Figure 3.2 Roulette Wheel Algorithm

Consider an illustrative example with 5 BCells, in a BCell population. Let us assume that the first BCell has the fitness 0.2, the second one has 0.7, third has 0.7, fourth has the fitness 0.9 and finally the fifth one has the fitness 0.4. Firstly, a fitness table is constructed from sum of each B cell.

Table 3.2 Fitness Table

Fitness Table		
0	Fitness of First B-Cell	0.2
1	Sum of First Two B-Cells' fitness	0.9
2	Sum of First Three B-Cells' fitness	1.6
3	Sum of First Four B-Cells' fitness	2.5
4	Sum of All B-Cells' fitness	2.9

After that a random number is created between first and last values of fitness table (Table 3.1). Two integers are created, one of them is for the first element of the wheel, and second one is created for the last element of the wheel. First number is set to zero, and last number is set to 4, then an integer that has a value of 3, which is the middle row of the Fitness Table. Middle number is calculated as half of the summation of the first and the last row. If the fitness number of the middle element is smaller than the random fitness number, then the first number is set to the middle number, else the last number is set to the middle number. This procedure is continued as is.

Assume that the random variable is equal to 1.8. Then, since the fitness in the third row is equal to 1.6, first number is set to middle number which is 3, the row number of the fitness 1.6.

The variable named middle number is set to (first+last)/2 which is (3+5)/2=4. Since the difference between first and last rows is not equal to one, the last number is set to middle number which is 4. Now, since the difference between first and last numbers are one, last number is returned from the function, and the fourth B-Cell which has a fitness of 0.9 is selected from the population, and put into sub population. After selection of B cell subpopulation, the selected cells are called parent B cells. Each of these cells is selected and produces mutated B cells. Child B cells are mutated randomly. Five different mutation operators are used as in antibody mutation but different than the case in antibody mutation.

The difference in these mutation operators is that they work for the zero to five scale.

(1) Multipoint mutation: Random numbers of bits are changed. A random number is generated and this is set to randomly selected bits.

B cell: 532331004210411321

Mutated B cell: 232331304210411311

Random number: 3

(2) Substring generation mutation: From a random start bit to an end bit, a substring is created from B cell string. Bits in this substring are regenerated randomly between 1 and 5.

B cell: 532331004210411321

Mutated B cell: 532125351200411321

Start Bit: 4

End Bit: 11

(3) Substring conversion mutation: From a random start bit to an end bit, a substring is created from B cell string. If the bit in the substring is zero it is changed to five, if it is one it is changed to four, if it is two it is changed to three, if it is four it is changed to one, else if it is five it is changed to zero.

B cell: 532331004210411321

Mutated B cell: 533224551305111321

Start Bit: 3

End Bit: 13

(4) Substring reversion mutation: From a random start bit to an end bit, a substring is created from B cell string string. The substring is reverted.

B cell: 532331004210411321

Mutated B cell: 534012400133211321

Start Bit: 3

End Bit: 13

(5) Substring copy mutation: From a random start bit to an end bit, a substring is created from B cell string. The substring is copied from a random starting bit.

B cell: 532331004210411321

Mutated B cell: 534012400133211321

Start Bit: 6

End Bit: 16

Copy Location: 2

After mutation, newer B cells' affinities are measured by using stimulation threshold formula shown in Equation 3.3. If they have affinities higher than a certain value these B cells are permitted to die. The remaining ones are added to the population. After generation of new B-Cells, a positive selection algorithm in which the system looks for B-Cells that have better affinity measures with the antigen is applied. B-Cells that have smaller similarity measures undergo apoptosis (programmable cell death), so that overall similarity of the new population is greater than the previous one. After positive selection algorithm B-Cells are exposed to self molecules, this process is called negative selection algorithm. B-Cells that are matched each other with a predefined level of affinity are permitted to die. The negative selection algorithm in human immune system protects body from cancer. In artificial immune system this algorithm permits new recommendations to be made by the system. We have to keep in mind that our aim is not to find the best solution. Our aim is to find a proper recommendation for the user. After negative selection algorithm, a predefined numbers of BCell is selected from the population is forming new BCell population of the last TCell. These BCells correspond to new recommendations.

3.3.2 Learning Stage

The learning stage of the immune system corresponds to the primary response of the natural immune system. The learning stage runs after the user logouts from the system. A TCell is generated using the demographic data of the user and a TCell population is formed by using the former users' demographic data, as in the classification stage.

The learning stage shown in figure 3.3 is the same as the classification stage algorithm (Figure 3.1) until the BCell creation phase, shown in the first top yellow

rectangle. Since the user has rated the items, the BCell is filled with this data. The differences between learning and classification are shown as yellow in Figure 3.3. Instead of stimulation threshold this time, a similarity measure is calculated between remaining B cells and the B cell created from current user (as shown in the second yellow rectangle shown in Figure 3.3). Pearson measure is used as fitness function.

Pearson Measure:

$$r = \frac{\sum_{i=1}^{n}(u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^{n}(u_i - \bar{u})^2 \sum_{i=1}^{n}(v_i - \bar{v})^2}}$$

u = current B-Cell string (visitor).

v = One B-Cell string from population (former visitors).

n = Total number of ratings of both users (current user and user selected from population) filled.

u_i = current order of visitor's ratings.

v_i = current order of former visitor's ratings.

u'= average order of visitors ratings.

v'= average order of former visitors ratings.

The remaining process is similar, but now the similarity measure is used instead of stimulation threshold.

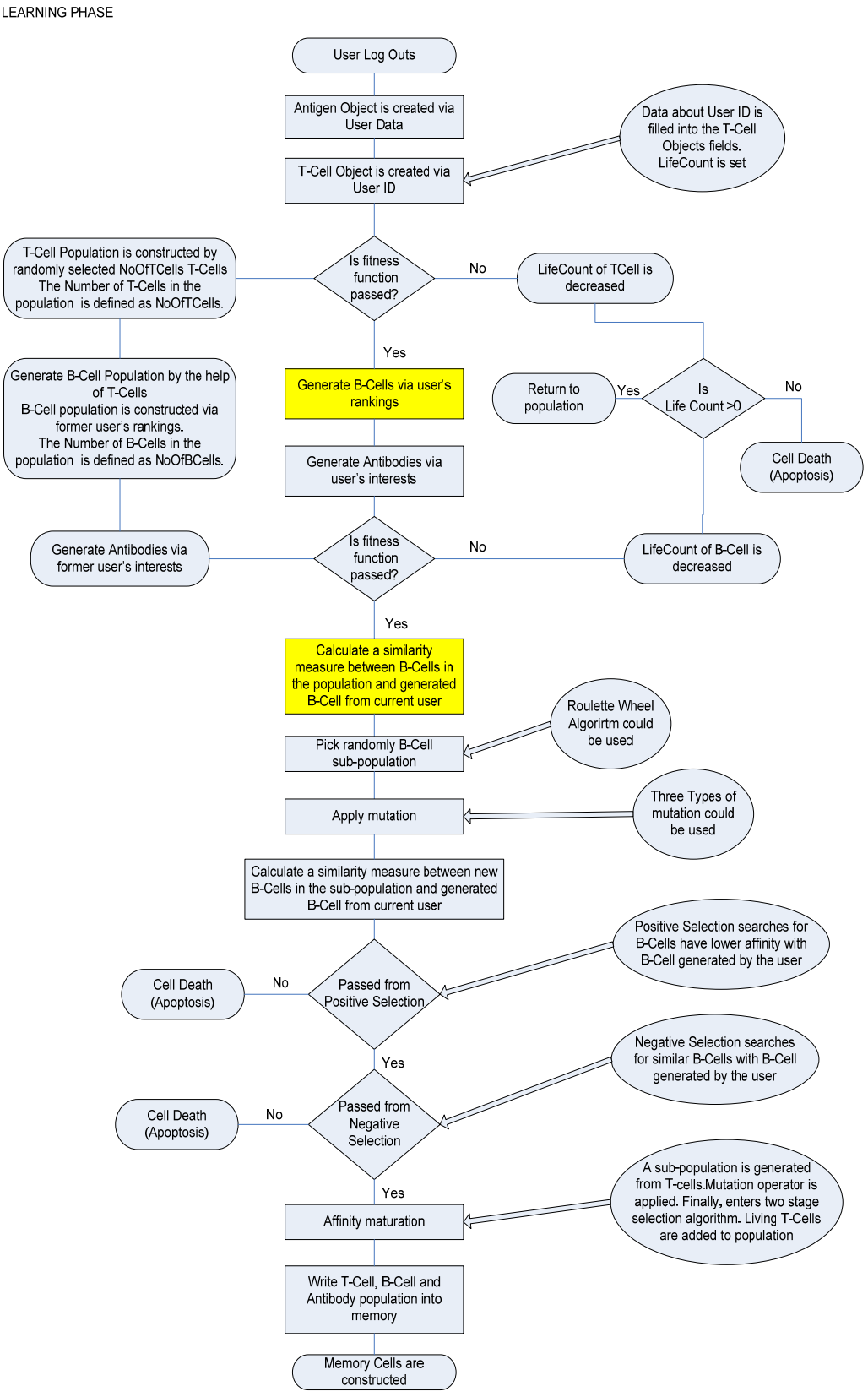

Figure 3.3 Learning Algorithm

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 ESGuide Immune Recommender System

The ESGuide project is designed as an intelligent city guide which allows visitors to tailor a tour from recommended locations, services and context-aware information based on their interests. The ESGuide mobile tourist guide application is composed of two components: Recommender agent and a map agent. The recommender agents use artificial immune system algorithm to deal with the recommendations. The map agents help the user to find the desired locations through the tour.

Figure 4.1 Welcome Screen

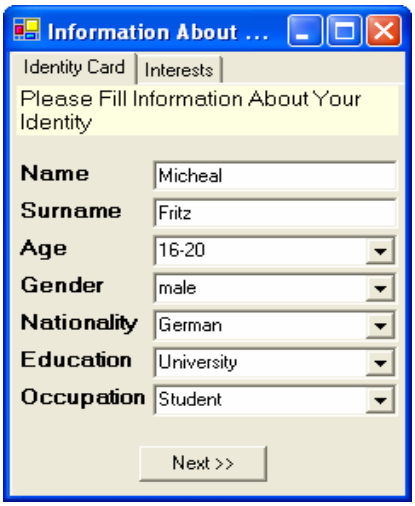

Figure 4.2 User Identity Screen

For a demonstration of how ESGuide works, lets assume a german boy visiting Eskişehir, (Turkey) aged of 18, educating in the university. Assume that this boy has an interest on American History, I.and II. World Wars in the history, enjoys playing

soccer, basketball and american football. This german boy also likes to eat spaghetti and fast food. He is interested in reading action, science fiction and horror books, and likes watching animation, horror and comedy movies.

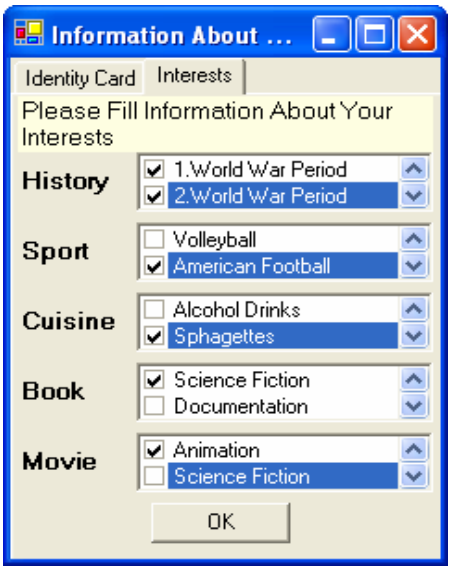

Figure 4.3 User Interests Screen

Figure 4.4 Main Screen

When the user starts the application, in the first screen a welcome message is displayed (Figure 4.1). After the user clicks the enter button in the screen, User information screen is displayed (Figure 4.2). This screen is composed of two parts. First part is the identity card, in this section the user has to enter his/her demographic information. He/she has to fill name, surname, age, gender, nationality, education and occupation. The German boy should fill this part as shown in Figure 4.2. In the second section user interests should be filled by the user. These interests include history, sport, cuisine, book and movie preferences of the user. This data is stored in a XML file named user. The boy should check the fields American History, I.and II. World Wars in the History section, soccer, basketball and american football in the sport section, sphagette and fast food in the cuisines section, action, science fiction and horror books in books section and animation, horror and comedy from movies section as shown in Figure 4.3.

Figure 4.5 Location Types Shown in the Map Screen

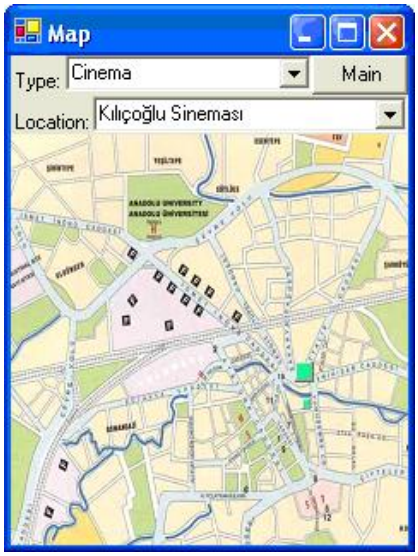

Figure 4.6 Overall City Map

After the user fills the identification card and interests, the main screen is shown (Figure 4.4). In this screen the user can click on map, places to go, things to do or logout buttons. Assume that the user clicks on the map button shown in Figure 4.4. After user clicks on the map button, map screen is shown to the user (Figure 4.5). In our implementation we have used the map of Eskişehir. The user can select different kinds of locations such that hotel, cinema, shopping, restaurant and pubs, historical places, sport and emergency (Figure 4.5).

Figure 4.7 Map Screen

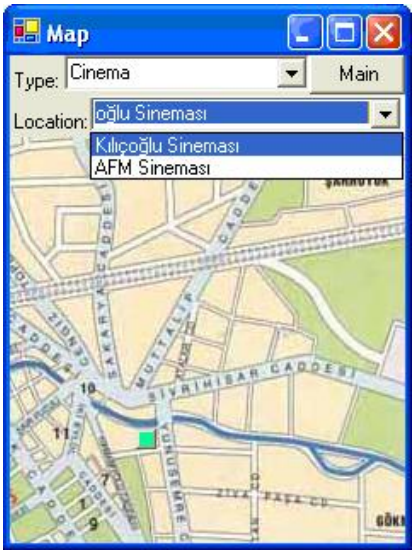

Figure 4.8 Locations Shown in the Map Screen

Assume that the user select cinema as the location type. The location named Kılıçoğlu sineması is shown to the user. Different types of locations is shown in different colors, such that in Figure 4.7 Hotel is shown as red, and in Figure 4.6 cinema is shown as light green. As shown in Figure 4.8 by clicking the combobox user can select different cinemas. The user also zoom out to see the overall city map, by clicking the right mouse button. To view a desired location the user should click on the left mouse button on the city map.

Figure 4.9 Kılıçoğlu Sineması Selected from Map Screen

Figure 4.10 Location Information Screen

Let us assume that the german boy selects location type as hotel, and selects Şale Otel from the combobox (Figure 4.7). Location information screen is (in Figure 4.10) shown to the user. In this screen opening and closing time of the hotel, description and price of the hotel is shown to the user. User can rate the place by clicking the stars in the please rate section. In Figure 4.10 the Şale Otel is not rated, and in Figure 4.11 Kılıçoğlu Sineması is rated as 4 stars, which means the cinema is known previously by the user who likes it. After the user should press GO button to return to the map in Figure 4.6. The user may go to the main menu by clicking "Main Button" in the map screen (Figure 4.6). By clicking the "Places to Go" button in this screen (Figure 4.4), locations screen is shown to the user (Figure 4.10). In this screen (Figure 4.10) the information about the location is presented to the user. This information is similar with the one presented in location information (Figure 4.12). But in this screen all the location in the same type is shown to the user. User can select the location by clicking to the "Select Button", or return to the main menu by clicking the "Back Button".

Figure 4.11 Rated Location

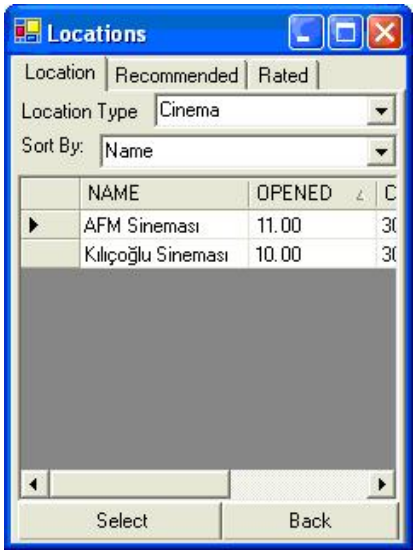

Figure 4.12 Locations Screen

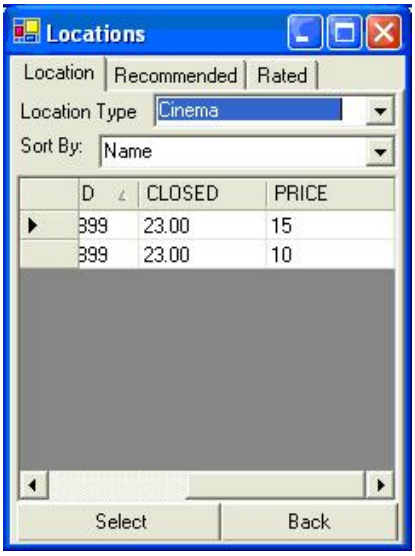

Figure 4.13 Using the Scroll

Figure 4.14 Selecting Different Types of Places

By using the horizontal scroll in Figure 4.13 the user can see the closing time and price of the location. The user can select different kinds of location as shown in Figure 4.14. At the top of this screen three tabs are shown: Location, recommended and rated. By clicking recommended tab, user can see the recommended places (Figure 4.15) generated by the recommender agent which is going to be presented in later parts of this chapter. In figure 4.15 seven different places are recommended to the user. Migros and Yunus Emre is the place that the user can like most, and Buda

Bar seemed to be not liked by the user. Let's assume that our german boy does not like these recommendations, i.e. he does not want to go to hotel, so he can press the "Decline Button", and the system will show him a new recommendation set (Figure 4.16). Now, Şale Otel has disappeared from the list. In this situation, when the user clicks on the decline button, lifecount of the BCell that generates this recommendation set, and the lifecount of TCell that generates this BCell will be decreased by one. Similarly, if in Figure 4.15 the user clicks on the "Accept Button" the recommender agent will increase the lifecount of the BCell that generates this recommendation set, and the lifecount of TCell that generates this BCell by one.

Figure 4.15 Recommended Places

Figure 4.16 New Recommendation Set

Let us say that the user clicks the "Accept Button", and accepts the recommendations then the "Accept" and "Decline" buttons disappear (Figure 4.17). If the user clicks on the "Rated Tab" on the top of this screen, the locations that have been previously rated by the user appear. Since the german boy rated Kılıçoğlu Sineması with 4 stars, and just looked on the Şale Otel's descriptions, but not rated this hotel, the rate is seen as zero (which means watched but not rated). After looking at the rated elements, the user can now click on the location bar, and then click on the back button (Figure 4.13). After clicking the back button the user enters to the main screen which is illustrated in Figure 4.4.

Figure 4.17 Accepted Recommendations

Figure 4. 18 Rated Locations

At that time, assume that the german boy starves and wants to see the cuisines in the application. The boy should click on the "Things To Do" button from the main screen (Figure 4.4). After clicking on this button "Things To Do" screen is shown to the user (Figure 4.19). In this screen, the user can click one of the five buttons including "Eat and Drink", "Watch Movie", "Buy a Book", "Main Menu" and "Log Out".

Figure 4.19 Things To Do Screen

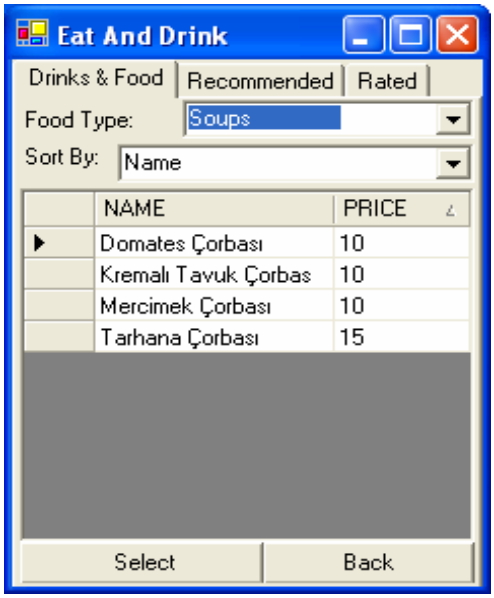

Figure 4.20 Cuisines Screen

Since the boy wants to eat something, he should click on the "Eat and Drink Button". After clicking on this button, Cuisines Screen (Figure 4.20) is shown to the user. Assume that the german boy wants to drink a special soup whose name is "Tarhana Çorbası", he can select it from the list (Figure 4.21), and assume that he likes that soup after that he eats it and gives 5 stars to that soup (Figure 4.22).

Figure 4.21 Tarhana Soup is shown to the user

Figure 4.22 Tarhana Soup is rated

Like in the locations part, recommended (Figure 4.21) and rated cuisines (Figure 4.22) can be seen by clicking the tabs on top of the cuisines screen (Figure 4.20). Similarly, in the "Things to Do Screen" shown in (Figure 4.19) the user can click on the "Watch Movie" to select a movie (Figure 4.23), or "Read a Book" to select a book (Figure 4.24).

Up to now user's actions and screens in the ESGuide tourist guide aplication are represented. Now, it is time to discuss how the recommendations are generated by the system.

4.1.1 Recommender Agent

Recommender agent uses artificial immune system architecture to make recommendations to the user while in mobility, and especially uses the immune system's pattern classification ability to tailor the tour to the user's own interest and personalize it.

The recommender agent consists of artificial cells that recognize the antigen (visitor) coming into the system. When the user enters the system, the system is stimulated and gives a secondary response, which means the classification algorithm is applied to the user to initiate the process of making recommendations.

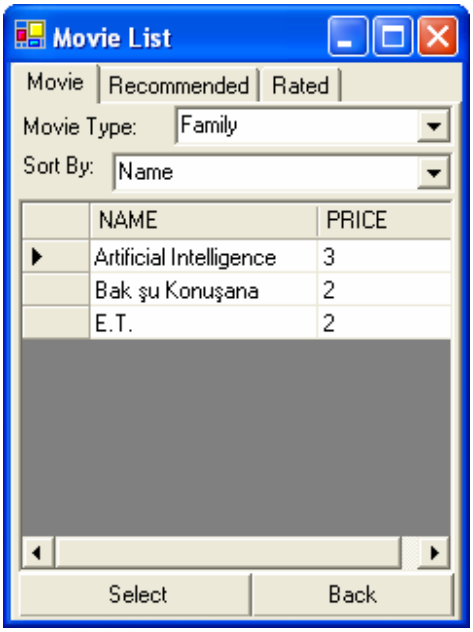

Figure 4.23 Watch Movie Screen

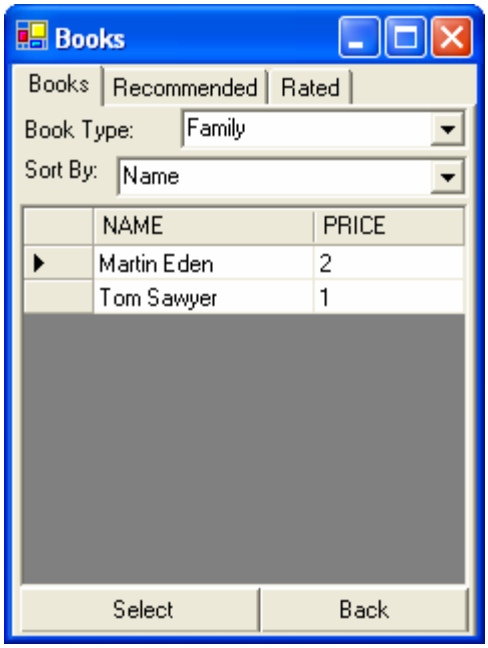

Figure 4.24 Read a Book Screen

Returning back to our german boy scenario, when he enters into the system and fills the data about himself as shown in Figure 4.3, and presses the OK button, the recommender agent creates an antigen object, that is the representation of the user in an immune system. Antigen object is composed of TCell part which is corresponding to the demographic data, BCell object constructed by the user ratings and antibody objects are created by the interests of the user. Then, antigen is ingested by the TCell object which means the demographic data of the user (Table 4.1). Since the age of the German boy is 18 which is between a range of 16-20, which is the second entry in Figure 4.25, the age field in the TCell shown in Table 4.1 is encoded as "000000000010" where 1 is placed on the second bit of the array. Since the german boy is educating in the university (which is the 5th row of Figure 4.26), the education field of the TCell shown in Table 4.1 is encoded as "00010000". Similarly Nationality, Occupation and Gender are encoded by looking the Figures 4.27, 4.28 and 4.29 respectively.

	ID	AGE_INTERVAL
	1	10-15
	2	16-20
	3	21-25
	4	26-30
	5	31-35
	6	36-40
	7	41-45
	8	46-50
	9	51-55
	10	56-60
	11	61-65
	12	66-70

Figure 4.25 Age Table

	ID	NAME
	1	Primary School
	2	Secondary School
	3	Military School
	4	High School
	5	University
	6	Master
	7	Ph D.
	8	Technical School

Figure 4.26 Education Table

	ID	NAME
	1	Turkish
	2	English
	3	German
	4	Italian
	5	Russian
	6	French
	7	Greek
	8	Bulgarian
	9	Japan
	10	Chinese
	11	American

Figure 4.27 Nationality Table

	ID	NAME
	1	Engineer
	2	Worker
	3	Teacher
	4	Doctor
	5	Student
	6	Architect
	7	Designer
	8	No Work
	9	Other

Figure 4.28 Occupation Table

	ID	NAME
	1	male
	2	female

Figure 4.29 Gender Table

After that a TCell population is constructed by the artificial immune system, the fitness score of each TCell with population is calculated including the TCell generated from the demographic data of the boy. The count of TCells in the TCell population is 100, so the ID of the TCell is set to 101 (Figure 4.1). LifeCount of the TCell is set to TCELL_LIFE_COUNT which is 100 in this demo scenario. The structure of one of the TCells in the TCell population can be seen in the Table 4.2. this cell is generated from an english girl whose age is between 16-20, educating in

secondary school, working as a blue collar worker. Since the demographic data of the german boy and english girl are very different, the fitness score between them is 0.285 which is near zero, as expected.

Table 4.1 Structure of TCell Object

TCell Object	
ID	101
Fitness	0
Age	000000000010
Education	00010000
Nationality	00000000100
Occupation	0000010000
Gender	01
LifeCount	100

Table 4.2 Structure of TCell Taken from the TCell Population

TCell Taken from the Population	
ID	1
Fitness	0.2857143
Age	000000000010
Education	00000010
Nationality	00000000010
Occupation	0000000010
Gender	10
LifeCount	0

Subsequently, BCells are generated by the TCell. For each different kind of item different types of BCells are generated such as placesBCell is created for locations, movieBcell is for movies, bookBCell is for books, cuisineBCell for cuisines. To illustrate the structures of the BCells we will show that of the placesBCell as in Table 4.3. Since the BCell is created newly, its ID is zero and locations field which

determines the rated locations is undefined. TCELL_ID of a BCell defines its creator TCell's ID in this case it is 101 (ID field in Table 4.1). LifeCount of the BCell is set to BCELL_LIFE_COUNT which is set to 50 by default. The locations field can be encoded by looking at Figure 4.30, such that as in Table 4.9, locations fields is encoded as "10004525000", where the first three bits from the right shows the Şale Otel, Kılıçoğlu Sineması and AFM Sineması given a rating of zero which means they have not rated, and Migros takes a rating of 5.

Table 4.3 Structure of placesBCell

BCell Object	
ID	0
TCELL_ID	101
Locations	Undefined
LifeCount	50
Stimulation	0

	ID	NAME
	1	Şale Otel
	2	Kılıçoğlu Sineması
	3	AFM Sineması
	4	Migros
	5	Oyak Alışveriş Merkezi
	6	Yunus Emre Köyü
	7	Gordion
	8	Atatürk Stadyumu
	9	Porsuk Spor Salonu
	10	Hayal Kahvesi
	11	Buda Bar

Figure 4.30 Locations Table

After that each BCell secretes antibody which consists of the user interests, the structure of the antibody generated from the german boy's interests is shown in Table 4.4. Since the antibody is generated newly, its ID is set to zero. The book field is encoded by using the Figure 4.31, since the german boy likes to read action, science fiction and horror books, the book field of the antibody is encoded as "000000010001010", the row numbers in Figure 4.31 show the position of the bits, starting from the right bit. For example, the second row in this figure is action, the fourth row is science fiction and the eight row is horror books, where rows in Figure

4.31 shows the bit positions of 1s in the array “000000010001010”. In Table 4.4 Cuisine, History, Movie and Sports fields are encoded by looking at figures 4.32, 4.33, 4.34, 4.35 respectively.

After generation of the antibody, BCells are generated by each of the TCells to be inserted in the TCell population. These BCells secrete antibodies against the interests part of the antigen, and a fitness score is calculated which analytical expression was given in section 3.3.1. Average antibody fitness is calculated after finding the fitnesses for each of the book, cuisine, history, movie and sport bit arrays. The structure of an antibody secreted by one of the BCells in the BCell population is shown in Table 4.5, and the structure of this BCell is shown in table 4.6.

Table 4.4 Structure of the Antibody

Antibody Object	
ID	0
TCELL_ID	101
Book	000000010001010
Cuisine	10000100
History	0011010000
Movie	0000001010000100
Sport	000000001011

ID	NAME
1	Family
2	Action
3	Animation
4	Science Fiction
5	Documentation
6	Drama
7	Fantasy
8	Horror
9	Adventure
10	Comedy
11	History
12	War
13	Romantic
14	Police
15	None

Figure 4.31 Book Interests Table

ID	NAME
1	Soups
2	French Meals
3	Fast Food
4	Cold Deserts
5	Hot Deserts
6	Drinks
7	Alcohol Drinks
8	Sphagettes

Figure 4.32 Cuisine Interests Table

ID	NAME
1	Ottoman Period
2	Seljuklu Period
3	Old Russian Period
4	USSR Period
5	American History
6	Chinese History
7	1.World War Period
8	2.World War Period
9	History of Aztecs

Figure 4.33 History Interests Table

ID	NAME
1	Family
2	Action
3	Animation
4	Science Fiction
5	Documentation
6	Drama
7	Fantasy
8	Horror
9	Adventure
10	Comedy
11	Musical
12	War
13	Romantic
14	Police
15	Short-Film
16	Serial

Figure 4.34 Movie Interests Table

ID	NAME
1	Soccer
2	Basketball
3	Volleyball
4	American Football
5	Tennis
6	Squash
7	Badminton
8	Swimming
9	Diving
10	Bowling
11	Chess
12	Snooker

Figure 4.35 Sport Interests Table

Table 4.5 Antibody secreted by the BCell in the BCell population

Antibody Object Secreted by the BCell	
ID	1
TCELL_ID	1
Book	010111111110101
Cuisine	11111111
History	1111111100
Movie	1111111101111110
Sport	010100010101
Average Fitness	0.615083337

After Antibody is secreted, the antibody is mutated to generate new antibodies, fitness score of each antibody is calculated, and an average fitness for an antibody is calculated as again shown in Table 4.5. An antibody and its mutated clone is shown in Figure 4.7, where bits of the fields book, cuisine, movie, history and sport is changed.

Table 4.6 Structure of BCell taken from BCell population

BCell Object	
ID	2
TCELL_ID	1
Locations	10004515400
LifeCount	50
Stimulation	0

Next, a stimulation threshold is measured by using the BCell and BCells gathered from the BCell population, one of the BCells in the BCell population is illustrated in Table 4.8. Then mutation is applied to these BCells, one of the mutated clone of a BCell is illustrated in Table 4.9. After that BCells are put into a two stage selection algorithm, and affinity maturation. These sections will be discussed in detail in Chapter 5. Finally the NoOfBCells (which is in that case five) of BCells that remain

after affinity maturation and is used to generate recommendations are given in Figure 4.15.

Table 4.7 Antibody secreted by the BCell in the BCell population

Antibody Objects Secreted by the BCell		
ID	0	0
TCELL_ID	1	1
Book	010111111111101	110100111110100
Cuisine	11111111	00101100
History	100111100	0010010000
Movie	101111111100111110	1111111101011010
Sport	010100011101	1010000110001

Table 4.8 One of the BCells in the BCell population after stimulation threshold

One of the BCells in the BCell population after stimulation threshold	
ID	2
TCELL_ID	1
Locations	10004515400
LifeCount	50
Stimulation	2.72411513

Table 4.9 One of the BCells in the BCell population after stimulation threshold

One of the BCells in the BCell population after stimulation threshold	
ID	2
TCELL_ID	1
Locations	10004525000
LifeCount	50

When the user logouts from the system, recommender agent works in the same way with the classification part. But this time locations part shown in the Table 4.3 is filled by the user ratings. And the system uses similarity measure instead of stimulation threshold.

4.2 System constants

Chapter 5 introduces discussions on test results There, in the tests, some constants will be changed to observe the reaction of the immune system, within the perspective of performance analysis. These constants are:

TCELL_LIFE_COUNT = Defines how many turns that a TCell can live in the immune system. If this number becomes zero the TCell will be deleted from the immune system.

TCELL_POPULATION_RATIO = This value defines percentage of the TCells will be taken from the database. If the value of this constant equals to TCell population is constructed with all of the TCells in the database.

BCELL_LIFE_COUNT = This constant defines how many turns that a BCell can live in the database. If this number becomes zero the BCell will be deleted from the database. If all of the BCells of a TCell is deleted from database, this TCell also be deleted from the database.

SUB_BCELL_POPULATION_RATIO = The ratio of the subpopulation of BCells, that will be created after calculation of BCell fitnesses.

ANTIBODY_POPULATION = This constant defines the number of antibodies that will be created by mutation of the original antibody belonging to the BCell .

ANTIBODY_SUBSTRING_GENERATION_MUTATION_RATIO,

ANTIBODY_MULTIPOINT_MUTATION_RATIO,

ANTIBODY_SUBSTRING_REVERSION_MUTATION_RATIO,

ANTIBODY_SUBSTRING_CONVERSION_MUTATION_RATIO,

ANTIBODY_SUBSTRING_COPY_MUTATION_RATIO = These five constanst will designate the ratios that antibody population will be mutated by different kinds of mutations. Total number of these constans must be one.

ANTIBODY_MIN_FITNESS = If average antibody of a BCell is lower than that value, BCell will be permitted to die.

PLACES_BCELL_MIN_FITNESS = In positive selection phase, the BCells fitness value should be greater than that constant, otherwise BCell will die.

PLACES_BCELL_MAX_FITNESS = In negative selection phase, the BCells fitness value should be smaller than that constant, otherwise BCell will die.

NoOfBCELLS = This value determine how many BCell that a TCell generate, also this number defines the number of recommended item lists.

We now concentrate on testing our system.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Generating Test Scenarios

Our tests include four different situations.

The first situation is initiated by a database that is formed with one class of users. Then, an antigen is constructed from a user that belongs to the same class as the one in the database. In the second situation, an antigen is constructed from a class totally different than those in the database. In this stiuation the database contains one class of users, with different profile than the new visitor entered into the system. The aim of the first and the second tests is to observe the behavior of the artificial immune system against a well known object, and an object totally foreign to the prior knowledge in the system. In the third and fourth situations, instead of a single visitor, a group of users enters into the system. A group of antigens which means a group of users are put into the immune system and the reaction of the immune system to these antigens are observed. In the first and second situation, the behaviors of the system is observed against a single antigen, while in the third and forth situations system behaviors are observed against a group of antigens.

The difference between third and fourth test is that in the former case the group of antigens are the same class with the cells in the immune system, while in the fourth test the antigens are unobserved class of antigens.

5.2 Test Situation 1: An antigen put in an immune system, constructed with similar class of cells

Let's assume that a Chinese man, whose age is between 61-65 years old, does not have a work but has a PhD. goes to Eskişehir as a visitor. He is interested in the II. World War period. He likes sports including soccer, volleyball, tennis, badminton,

diving and chess. He is interested in family, animation and police books. He likes to drink alcohol, and likes watching serials. This man and his friends which have similar properties of age, gender, nationality and occupation, came in Eskişehir and have used our ESGuide application so that our system has already learned their profiles, which are stored in the database. This group of users also has the same interests, and has given similar rating to the same locations. The demographic data of these users are encoded in the TCell portion of Figure 5.1, where the users' interests are encoded in the antibody portion of this figure and finally, the rated element are encoded as placesBcell field in the same figure 5.1. The number in this whole group of people is 100 which makes up the TCell population in Figure 5.1, and since the Change max. bits in Figure 5.1 is set to 1, only one attribute (i.e. age, gender, nationality, education, occupation, history interests, sport interests, book interest, cuisine interests, movie interests) of these people could be different than the Chinese man.

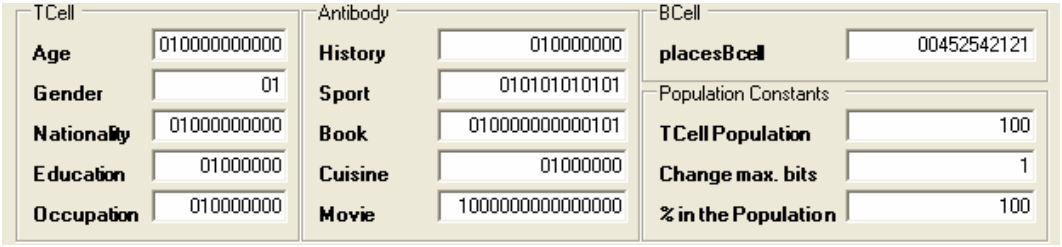

Figure 5.1 Immune Cells Constructing the Test Database

The constants defined in section 4.2 are set to values defined in Figure 5.2.
TCELL_LIFE_COUNT = 100; this means since the number of TCells in the artificial immune system is equal to the TCELL_LIFE_COUNT, any of the TCells are deleted from the database with the effect of TCells lifecounts decreased to zero.
TCELL_POPULATION_RATIO = 1; since this value is equal to one, all of the TCells in the database will be put on the TCell population.
BCELL_LIFE_COUNT = 50, the value is half of the number of TCells, so the BCells have a high possibility to die during training. But in the first situation, since the users are similar, none of the BCells will die. Since the lifecount of BCells that have low affinity is decreased by one, if BCELL_LIFE_COUNT is taken small then these BCells will die, and the recommendations presented to the user can adapt to

chances in trends very fast, but in this case the system is easily affected from the errors made in ratings.

ANTIBODY_POPULATION = 10, the value of 10 is appropriate for mutated antibodies, that their average fitness value determines the BCells which will be mutated. If this number is high in the order of twenties, then all of the antibodies will be formed after mutation, and they will be very different from the original one which means the users that have very different interests will pass the clustering. If this number is in the order of fives, then passed users from this clustering stage will be just user whose interests are very similar.

ANTIBODY_SUBSTRING_GENERATION_MUTATION_RATIO = 0.2

ANTIBODY_MULTIPOINT_MUTATION_RATIO = 0.2

ANTIBODY_SUBSTRING_REVERSION_MUTATION_RATIO = 0.2

ANTIBODY_SUBSTRING_CONVERSION_MUTATION_RATIO = 0.2

ANTIBODY_SUBSTRING_COPY_MUTATION_RATIO = 0.2; These numbers define which mutation operator is used. By giving all of those ratios 0.2, all of the operators are used.

ANTIBODY_MIN_FITNESS = 0.3, since our group of users have the same interests, setting this number to 0.3 makes nearly all people pass the clustering algorithm, and gives us a chance to observe BCells.

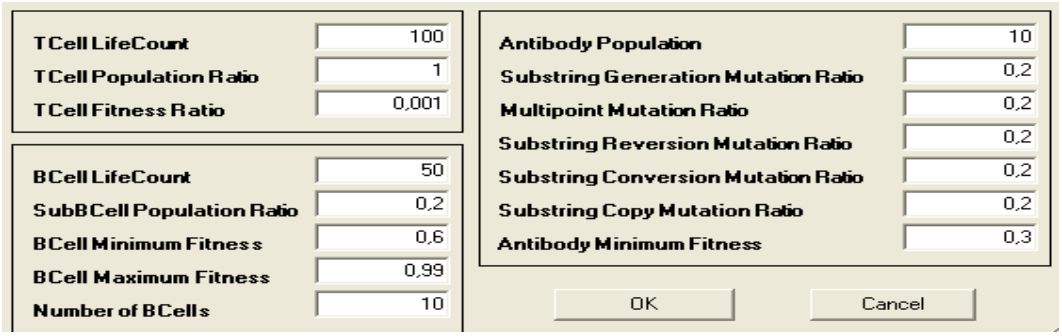

Figure 5.2 Test Immune System Constants

Then a new Chinese visitor whose name is Wang enters the system, possessing demographic data which are same as those of the former Chinese man. Wang additionally likes fantasy and short films in addition to serials. Also Wang has rated a few places such as Buda Bar with different ratings than that of the Chinese.

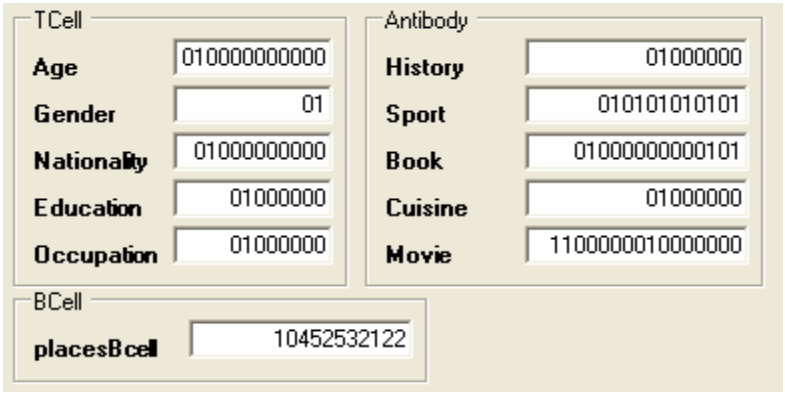

Figure 5.3 Properties of the User

Wang after rating the contents, logs out from the system; An antigen object is created with the help of the fields shown in Figure 5.3. TCell part shown in this figure is constructed by encoding Wang's demographic data, antibodies are generated from Wang's interests, and placesBcell field shown in Figure 5.3 is constructed with Wang's ratings given to the locations. Then, TCell object is created by the encoded data of Age, Gender, Nationality, Education and Occupation shown in Figure 5.3. After TCell generation, a TCell population is created by using the demographic data of the former visitors whose properties are similar with our recent visitor. Then, fitness score between each TCells in the TCell population is calculated which is given in Figure 5.4. In this figure, the vertical axis of the graph shows the fitness scores , horizontal values shows the ID of each TCell in the TCell population. Since the population count was defined as 100 in Figure 5.1, the horizontal axis ends with a value of 100. The fitness value curve spans the range between 0.7 and 1.0, its average value being between 0.81 and 0.82. In figure 5.4, higher fitness values means more similar users, and a fitness score of 1 means that these two users have the same demographic data, like Wang and the former Chinese man. Since the former users visited the city have similar demographic data with our recent visitor Wang, scores are higher even having the values close to 1. The average value of fitness scores that is between values 0.81 and 0.82, obeys our assumptions, since the new visitor has similar data with prior users.

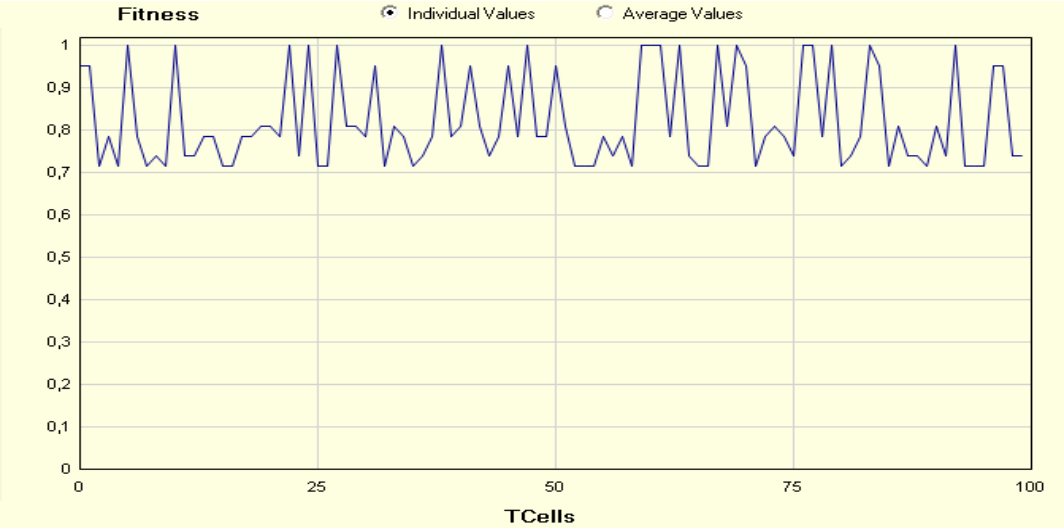

Figure 5.4 Fitness score of TCells in the TCell Population

Following this, the BCell population is constructed by the TCells in the TCell population. The BCell created by the ratings of our visitor is shown in Figure 5.3. These BCells will secrete antibodies which are constructed by the user interests. In the Immune system former users constructs a BCell population with the mutation of BCell shown in Figure 5.1. Former BCells constructed in the immune system can differ just one bit from the BCell shown in Figure 5.1, since the change max. bits value is set to 1 in Figure 5.2. Since the rated locations and their ratings are similar, expected fitness values between the BCell and BCells in the BCell population is closer to 1. In Figure 5.5, vertical axis shows the similarity values of each BCell in the population with respect to the BCell created from the recent visitor's rated locations. Each BCells are shown as on the horizontal axis. The maximum value for the horizontal axis is equal to 1000 which is the population count of the BCells. In Figure 5.5, since the rated items are similar, nearly all of the fitness values lie between 0.8 and 1 which is a very high value. Figure 5.5 is taken after secretion of antibodies, since the similarity drops to zero in some of the BCells shown in this figure, this is because if the BCell can not recognize the antigen by secretion of antibodies, then this cell is permitted to die.

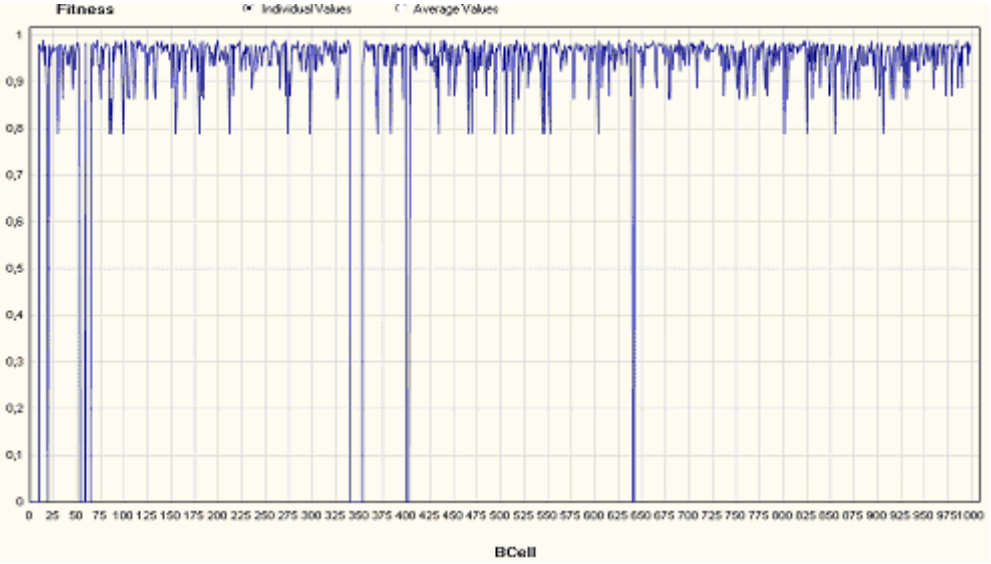

Figure 5.5 Similarity Measure calculated between BCell created from user rated locations, and the BCells created from former users rated locations.

Let's look at Figure 5.6, which shows the fitness values of antibodies. curve represents the fitness values between antibody which is created from the recent visitor's interests shown in Figure 5.3 and antibodies created from the former user's interests shown in Figure 5.1. The horizontal axis shows the fitnesses that belong to each antibody, where the vertical axis shows the number of BCells in the BCell population. Since 10 antibodies are generated for each TCell (which is shown in Figure in 5.2 as antibody population) the value equals to 1000, while TCell population is equal to 100 (shown in Figure 5.1 as TCell Population). Since the antibodies are mutated their fitness scores are vibrated between 0.2 and 0.8 which is shown in figure 5.6. We have defined the ANTIBODY_MIN_FITNESS thereshold value as 0.3 which is shown in Figure 5.2. So the BCells whose antibody fitness values are below 0.3 is permitted to die, and their fitness values become zero. In Figure 5.5, the fitness of BCells which drop to zero, has antibody fitness (shown in Figure 5.6) below the ANTIBODY_MIN_FITNESS value. If the user wants to change the importance of interests on recommendations, then he/she should adjust the ANTIBODY_MIN_FITNESS constant. If the user increase this constant, then interest becomes more important, since the BCells whose antibodies that can not pass this threshold value will be dead, the ratings that construct these BCells won't be shown in the recommendation set so more death is then expected with this higher

threshold. If the user decrease that constant, then importance of this constant decreases. It could be advised that, if the user enters into the system where former users had the same profiles, this constant can be increased. But if the users have different profiles, then this constant should be decreased, meaning that most of the BCells will be perished and recommendation quality will be decreased dramatically.

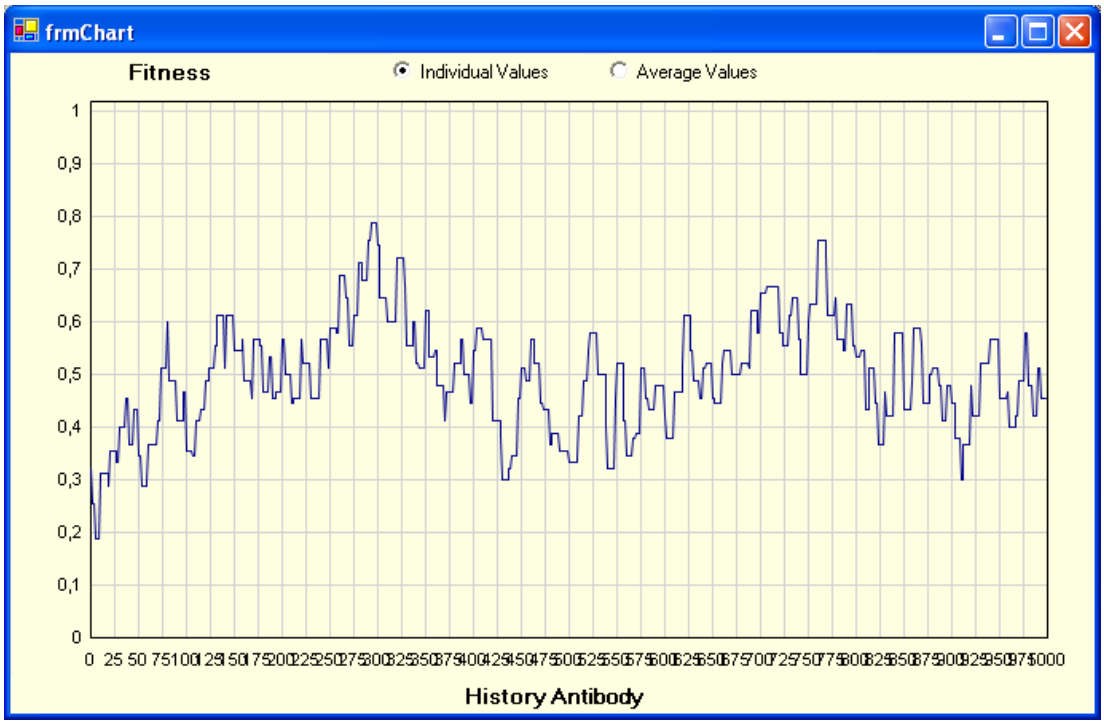

Figure 5.6 Fitness scores calculated between the visitor interests and former visitors interests.

After secretion of antibodies, the dead BCells whose similarity measures are equal to zero, are removed from the BCell population. The living BCells are formed as a subpopulation. Since the BCells that have a fitness of zero are removed from the population, the average similarity measure of the new BCell population (shown in figure 5.7) is greater than the previous one. In Figure 5.7, the vertical axis shows the similarity measures of BCells calculated between these cells and the BCell constructed from the visitor's rated locations. The horizontal axis in this figure shows the BCells in the new BCell population. The curve (in Figure 5.7) spans the range of 0.8 and 1 where in previous BCell population shown in Figure 5.5 this range spanned all the values range of 0 and 1.

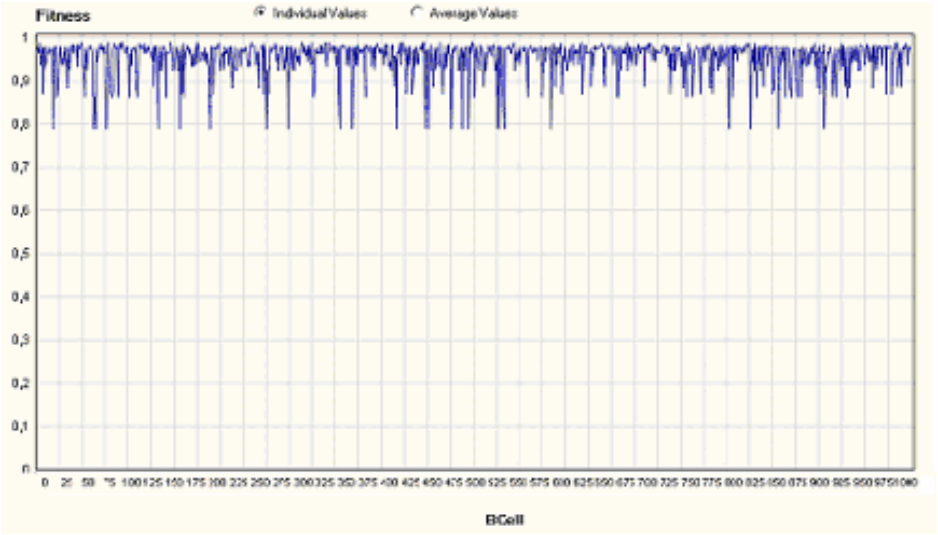

Figure 5.7 Similarity measures calculated between the visitor and former visitors' interests.

After calculation of similarity measures, a group of BCells are selected from the population, and a subpopulation is constructed. Then, mutation operator is applied to BCells in the subpopulation. Since, before the mutation BCells in the population was very similar to the BCell constructed from the new visitor's ratings, changing the bits of BCell with random ones will cause the decrease in average similarity measure of the overall BCell population (Figure 5.8), despite the increase in similarity measures of a few positively mutated BCells. If the BCells in BCell population was not so similar to the BCell constructed from user ratings, then the average similarity measure of the overall system would be increased.

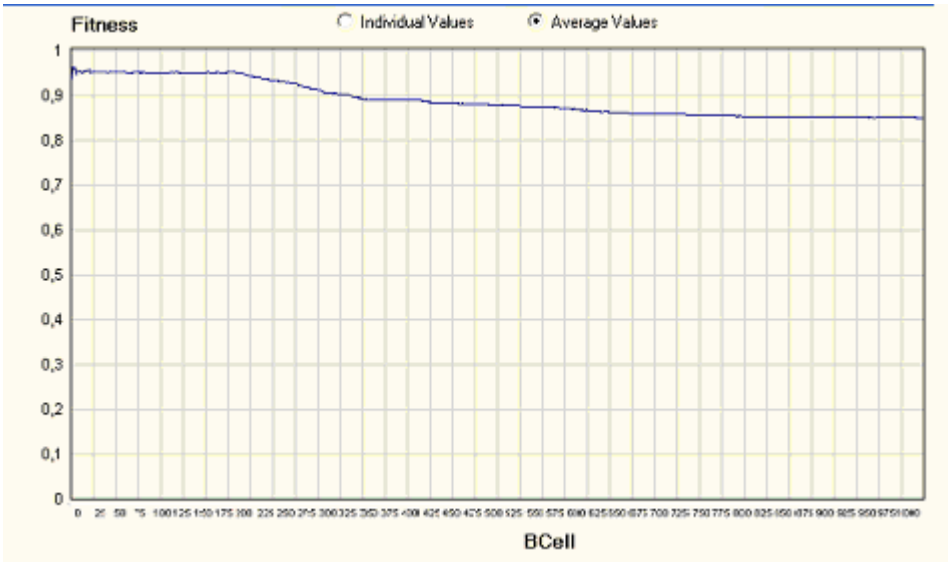

Figure 5.8 The average of similarity measures calculated after mutation.

The decrease in similarity measures of BCells can seen more clearly in Figure 5.9. Before mutation, the similarity measures lied between 0.8 and 1 (Figure 5.7), and after mutation these values lie between 0.6 and 1 (Figure 5.9).

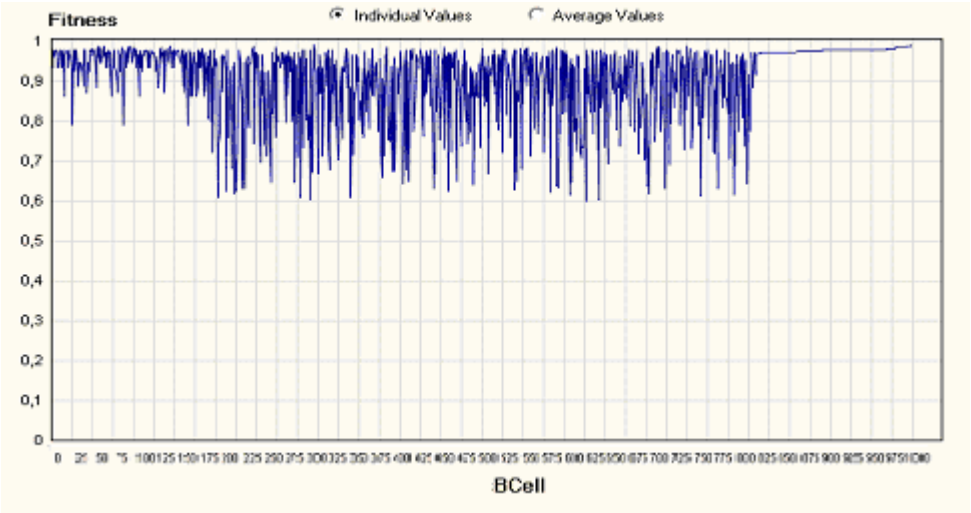

Figure 5.9 Similarity measures calculated after mutation of BCells.

After that process the BCells are put into a two level selection process, positive and negative selection. Two values in this stage is important: PLACES_BCELL_MIN_FITNESS, which defines the minimum threshold value for BCell similarity measures, that can live after the positive selection algorithm, and set to 0.6 (Figure 5.2). The second important constant in this stage is PLACES_BCELL_MAX_FITNESS which is set to 0.99, which determines the

maximum threshold value that similarity measure of a BCell in the subpopulation can exceed, if the similarity measure of the BCell exceeds this value, then the BCell is permitted to die, and is eliminated from the population. Since the similarity measures lie between 0.6 and 1 shown in Figure 5.9, which is between the threshold values PLACES_BCELL_MIN_FITNESS (which is 0.6) and PLACES_BCELL_MAX_FITNESS (which is 0.99), the BCells are not affected from this two selection process. But if the PLACES_BCELL_MIN_FITNESS was set to 0.8 previously, then BCells that have lower similarity measure than 0.8 is permitted to die, and the curve plotted in the figure 5.9 would be changed as the ones which it is plotted in figure 5.10.

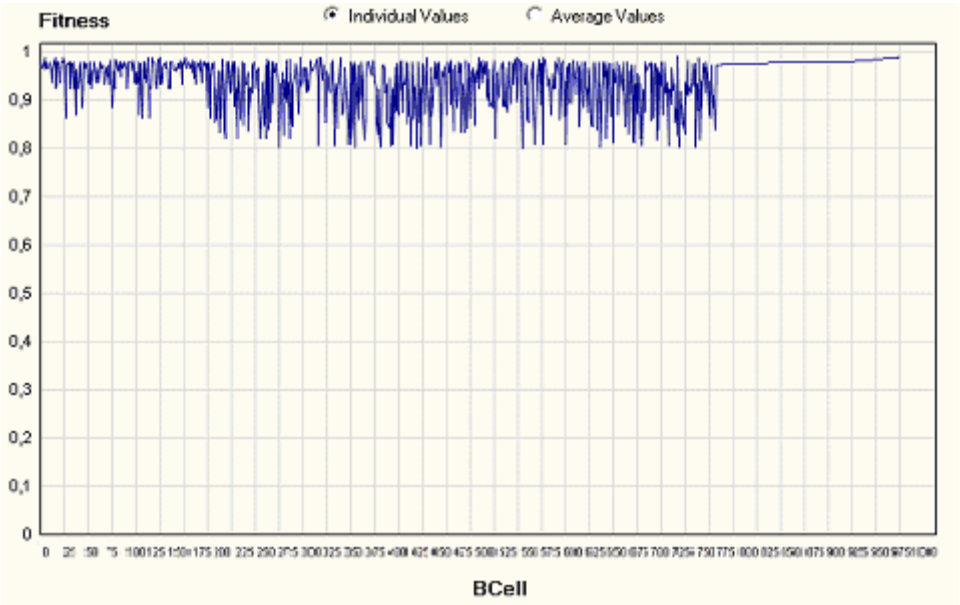

Figure 5.10 Similarities calculated after PLACES_BCELL_MIN_FITNESS is set to 0.8.

If the BCell population is constructed by similar BCells, PLACES_BCELL_MIN_FITNESS should be increased, but if these BCells are not similar then this value should be decreased to allow more BCells to pass the threshold. If PLACES_BCELL_MAX_FITNESS is equal to or near 1, then the recommendations will be made is very similar to the similar user's good rated items. In fact if a user rates the items and logouts from the system, and then re-enters the system again, if PLACES_BCELL_MAX_FITNESS, the artificial recommender system will recommend this user the items that he/she has already rated. This could

be boring for the visitor, so this constant should not be set to 1 in any conditions. Finally the recommendation BCells are constructed by the system, in Figure 5.11 individual fitness of each BCell is shown. It can be seen that fitness values lie between 0.8 and 1.0, and average fitness value for recommendation BCells are about 0.92 as expected, since the users have similar profiles with our visitor Wang. Then these BCells are decoded, and a recommendation list is created, and presented to the user (Table 5.1).

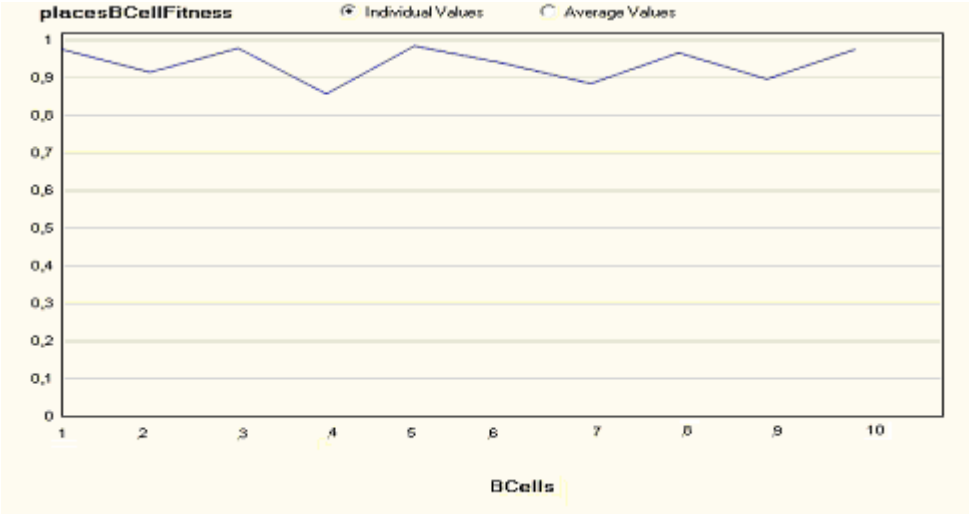

Figure 5.11 The fitness of BCells that are recommended to the user.

Table 5.1 Recommendation sets shown to the user.

Recommendation List										
Locations	Recommendation Sets									
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Şale Otel	0	0	0	0	0	5	0	0	0	4
Kılıçoğlu Sineması	2	0	0	0	0	4	0	0	0	1
AFM Sineması	4	4	4	4	4	0	4	4	4	3
Migros	5	5	5	5	5	5	5	5	5	5
Oyak Alışveriş Merkezi	2	2	2	2	2	2	3	2	0	4
Yunus Emre Köyü	0	5	4	5	5	5	0	5	5	5
Gordion	3	4	5	1	4	4	1	3	4	4
Atatürk Stadyumu	2	2	1	3	2	2	3	1	2	2
Porsuk Spor Salonu	1	4	1	1	1	1	4	1	1	1
Hayal Kahvesi	2	2	2	2	2	1	2	2	2	2
Buda Bar	1	3	1	4	4	1	2	1	1	1

5.3 Test Situation 2: An antigen put in an immune system that is constructed with different class of cells with this antigen

The immune database for testing the 2nd situation differs from the first situation only by the fact that, the new comer into the system has a profile totally different than the users in the test database. This new comer is a 31-35 years german woman, with an education up to the end of secondary school, and works in a totally different field that was listed in the occupations field in Figure 4.2 namely farmer, The woman has interested in Ottoman and Seljuklu period in the history, likes to read comics and history books, also likes watching family, action, comedy musical and war movies. She likes playing soccer, tennis, badminton, chess and snooker. She enjoys eating soups and french meals. She have rated the locations Kılıçoğlu Sineması with a rating of 1, AFM Sineması as 5, Migros as 4, Oyak Alışveriş Merkezi as 5, Gordion as 1 and Buda Bar as 5.

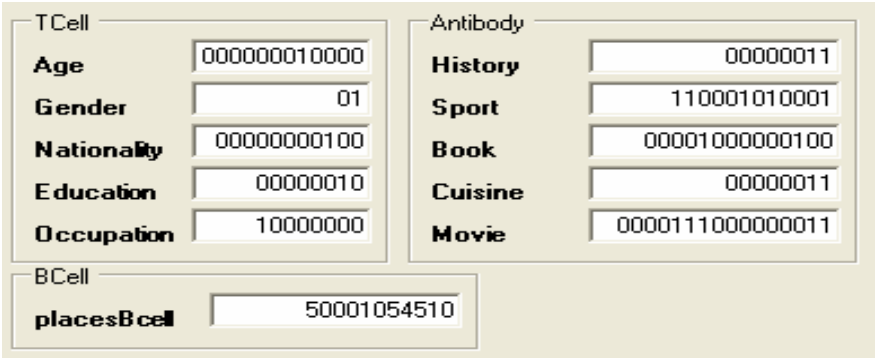

Figure 5.12 The properties of the user in the 2nd situation

After the user entered into the system, his demographic data is encoded to construct a feature vector named TCell (figure 5.12), and user interests are encoded to form a feature vector which is an antibody (figure 5.12), and finally a feature vector named BCell is constructed with encoded user ratings. After that a TCell population is constructed from the former users' demographic data that reside in the database. The fitness between TCell constructed from the user's demographic data, and TCells in the TCell population is calculated for each TCell in the population (Shown in Figure 5.13). In this graph the vertical axis shows the fitness values for each TCell in horizontal axis. Since demographic data of the user and the former users are different, the fitness values span the range of 0.35 and zero, and most of the fitness values are equal to 0.5, which is very close to zero. In the first situation fitness values of TCells (Figure 5.4) was close to 1 since the profile of the user and former users are very similar to each other.

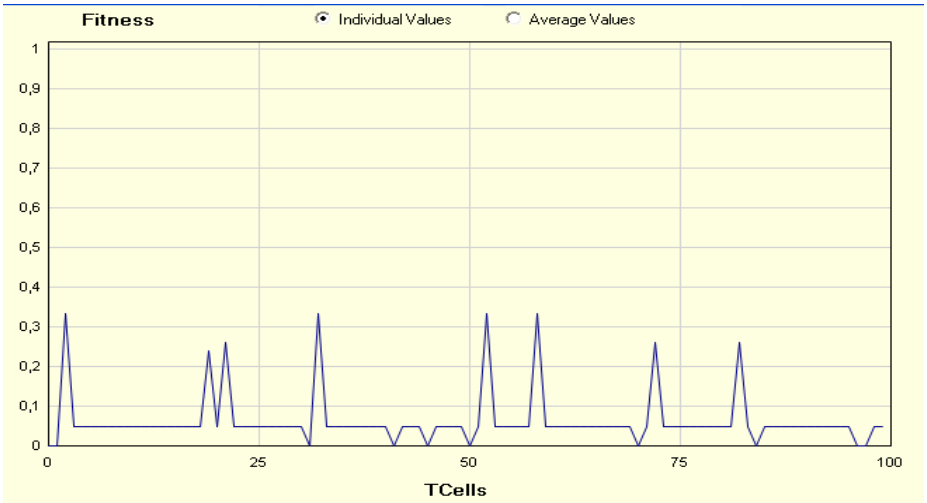

Figure 5.13 Fitness measures calculated between TCell and TCells in the population

Next BCell population is constructed by the TCells in the population. These BCells are constructed from the ratings of the items. In the test environment, we have discussed only the placesBCells which are constructed from the rated locations. In Figure 5.14, vertical axis shows the similarity measures, and the horizontal values determine corresponding BCells. Since the user's ratings are different than the ratings of the former users, the similarity measures are not as high as the ones in the first situation shown in Figure 5.5. The curve in Figure 5.14 is fluctuating because the BCells stored in the database are the mutated clones of in Figure 5.1 by changing one bit for each BCell (defined as Change Max. bits in the same figure). Majority of similarity measures of BCells spans a region of 0.3 and 0.7, where this region was between 0.8 and 0.1 in the previous test situation. Figure 5.5 and 5.14 together, clearly show that the similarity between two users are high, if the similarity measure between the BCells constructed from the ratings of these users is as high as 1.

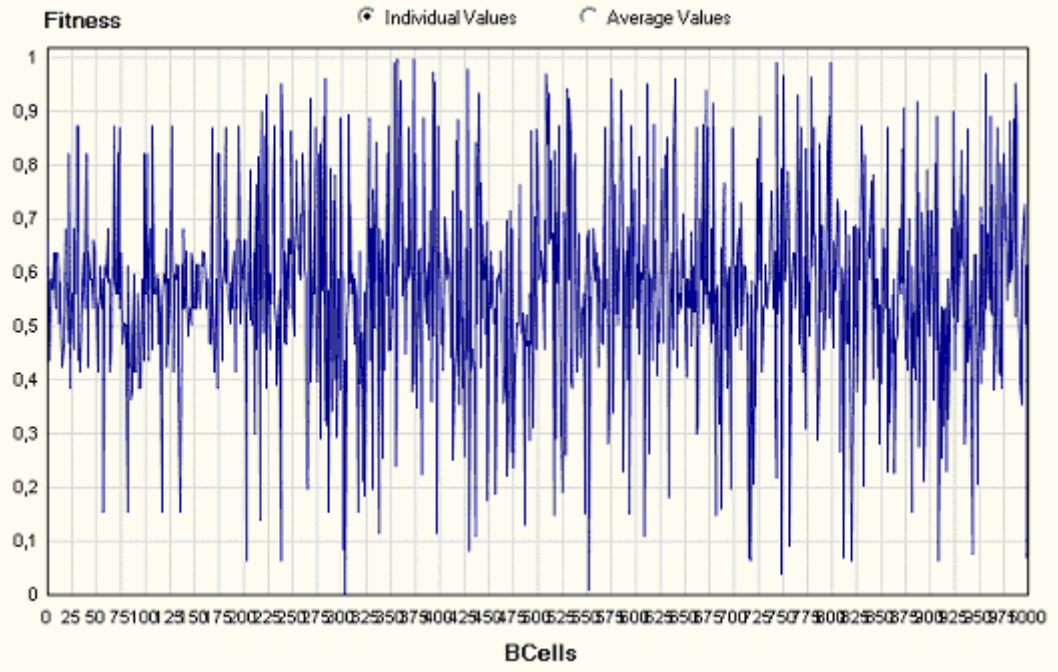

Figure 5.14 Similarity Measures calculated between BCell and BCells in the population

After calculation of similarity measures, the antibodies are secreted by the BCells. In this part of the algorithm, the similarity measure of the BCell, whose all of the antibodies (antibodies of a BCell are historyAntibody, movieAntibody, bookAntibody, sportAntibody and cuisineAntibody) can not reach

ANTIBODY_MIN_FITNESS will be set to zero, and this BCell is taken out of the BCell population. In figure 5.15, the cuisineAntibody that is constructed by the cuisine interests of user, and in Figure 5.16, the movieAntibody that is constructed by the movie interests of user are discussed. In both figures, vertical axis of the diagrams shows the antibody fitness values, and in the horizontal axis for Figure 5.15 the IDs of cuisineAntibodies are shown, and in Figure 5.16 the IDs of movieAntibodies are shown. Since ANTIBODY_MIN_FITNESS is set to 3, if one of the antibodies with the same ID reaches the threshold value, the BCell is not die, and return to the BCell population. In figures 5.15 and 5.16, a few antibodies have a fitness value below 0.3, and since these antibodies have different IDs, None of the BCells die after secretion of the BCells.

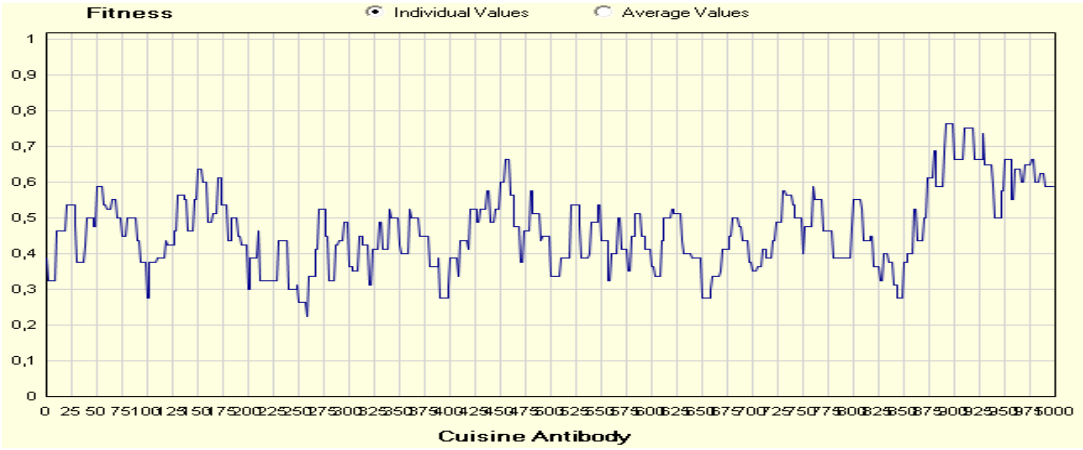

Figure 5.15 Fitness values of Cuisine Antibody

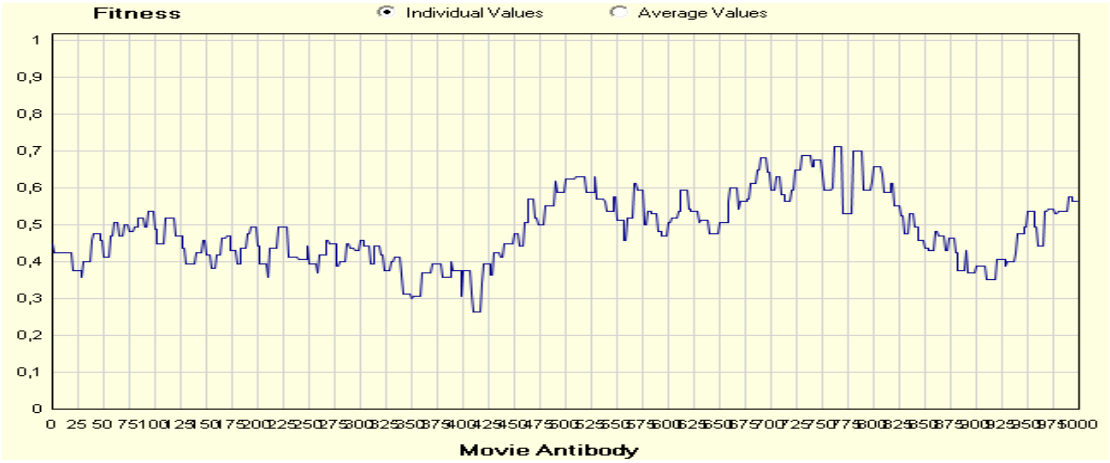

Figure 5.16 Fitness values of Movie Antibody

After secretion of antibodies, a groups of BCells are selected from the BCell population as we had done in the first situation. A subpopulation is constructed by these cells, and these cells are mutated. After mutation similarity measures of the BCells in the subpopulation are re-calculated (in Figure 5.17). It can be easily seen in this figure, the mutation operator has increased the spanning range of the curve to a range between 0 and 1, previously this range was between 0.5 and 9 as shown in Figure 5.14. In the first situation, the mutation operator decreases the similarity measures of BCells (Figure 5.9), but in the second situation, the mutation affected the spanning of the curve enlarging the interval which was between 0 and 1. This demonstration shows that the affect of mutation operator in similarity measures of BCells depends on the similarity of the users.

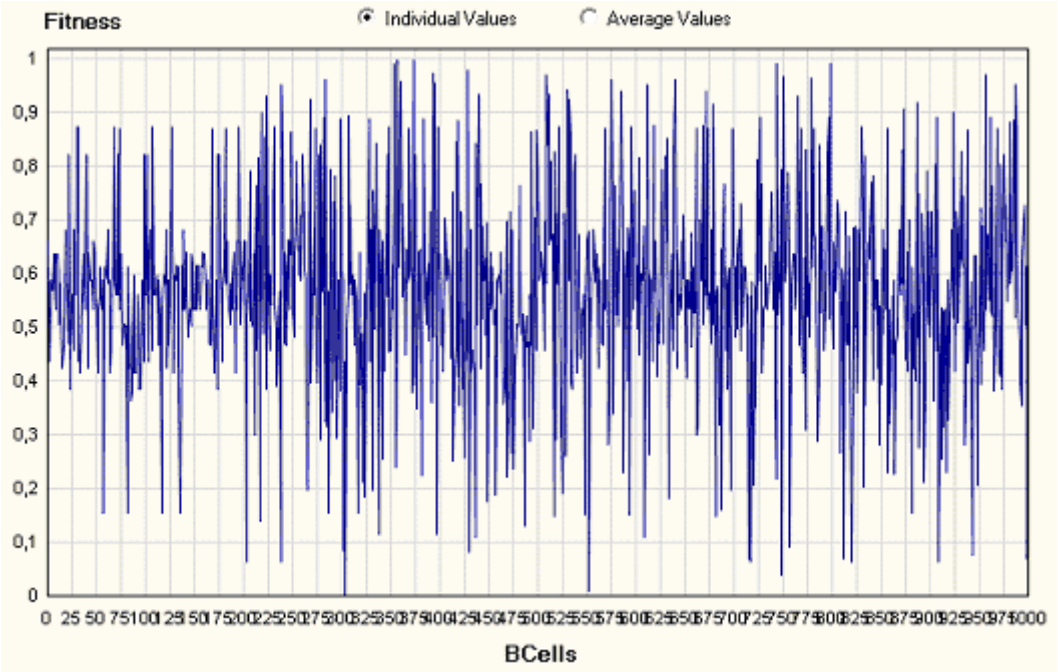

Figure 5.17 Similarity measures after mutation

After mutation applied on this subpopulation, BCells are put into two stage selection algorithm and then put into the BCell population, as in the first situation: positive and negative selection. As said in the first situation two values is important: PLACES_BCELL_MIN_FITNESS, which defines the minimum threshold value for BCell similarity measures, for them to that can live after the positive selection algorithm, is set to 0.6 (Figure 5.2). The second important constant in this stage is PLACES_BCELL_MAX_FITNESS which is set to 0.99, determining the maximum

threshold value that the similarity measure of a BCell in the subpopulation can exceed. If the similarity measure of the BCell exceeds this value, then the BCell is permitted to die, and is eliminated from the population. Different than the first situation, in the second situation the similarity measures lies between 0 and 1, so the threshold values affect the algorithm as shown in Figure 5.18. Since the BCells that have similarity below the PLACES_BCELL_MIN_FITNESS (which is 0.6), and above PLACES_BCELL_MAX_FITNESS (which is 0.99) are eliminated from the population, the curve in Figure 5.18 now spans the region of 0.6 and 0.99. If we compare Figures 5.10 and 5.18, setting the PLACES_BCELL_MAX_FITNESS value higher as 1, makes the recommendations more related to the user's ratings, and by decreasing this value, recommendations are more affected from the former visitors ratings, instead of user's ratings. If we examine both curve carefully, we can say that PLACES_BCELL_MIN_FITNESS affects recommendations' reliability. If we set this value near zero, then unreliable recommendations could be generated which are formed during mutation of the BCell.

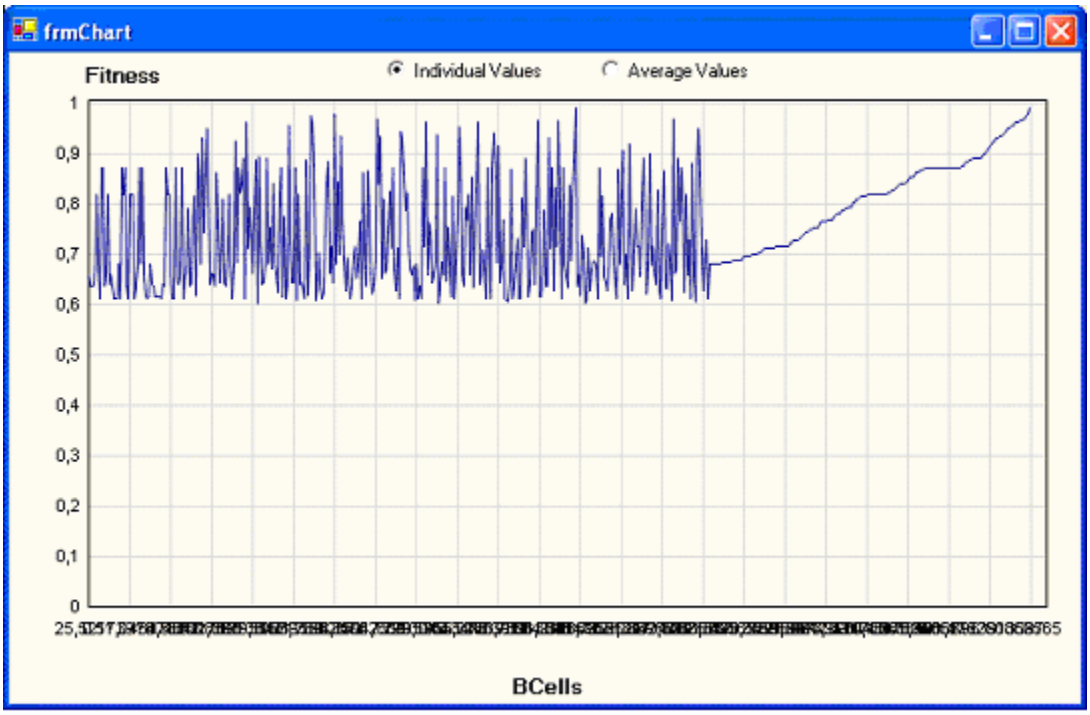

Figure 5.18 Similarity measures after positive and negative selection

Finally the BCells that will generate the recommendations are constructed by the system (Figure 5.19). These cells are selected from the population with the help of

the roulette wheel algorithm. It can be seen that similarity measures lie between 0.6 and 0.95, and average fitness value for recommendation BCells are about 0.8 which is smaller than the value found in the first situation which is about 0.92 in the first situation as expected, since the users have different profiles with our german visitor. To generate a recommendation set which is affected more from the recent visitors profiles, PLACES_BCELL_MAX_FITNESS should be decreased in the second situation. Then these BCells are decoded, and a recommendation list is created, and presented to the user (Table 5.2).

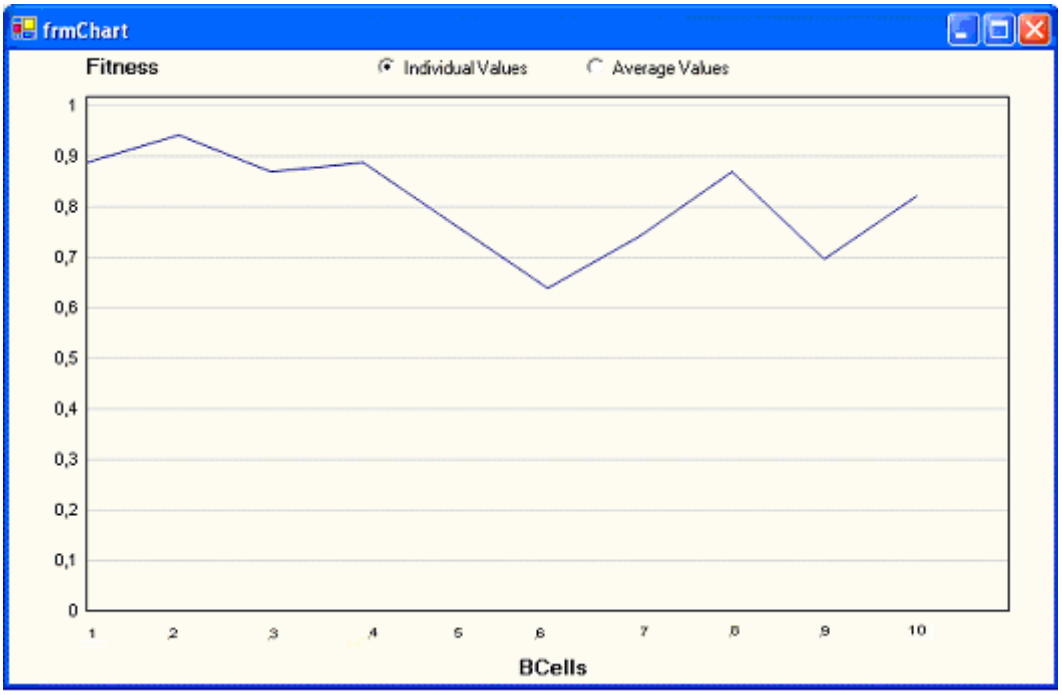

Figure 5.19 Recommendation BCell generated in the second situation

Table 5.2 Recommendation sets shown to the user in the second situation.

Recommendation List										
Locations	Recommendation Sets									
	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Şale Otel	2	0	0	0	0	0	0	5	0	0
Kılıçoğlu Sineması	2	0	4	0	0	0	0	5	0	4
AFM Sineması	1	4	2	4	4	4	4	1	4	4
Migros	0	0	1	5	5	0	5	0	1	5
Oyak Alışveriş Merkezi	3	2	1	1	2	2	2	4	2	2
Yunus Emre Köyü	0	5	5	5	1	5	5	0	1	5
Gordion	2	4	4	4	2	4	4	1	4	2
Atatürk Stadyumu	4	2	2	1	4	2	4	3	2	5
Porsuk Spor Salonu	0	1	1	3	2	1	1	4	1	4
Hayal Kahvesi	2	2	2	0	2	2	1	3	2	2
Buda Bar	1	1	1	1	1	1	1	1	1	1

5.4 Test situation 3: A group of antigen put in an immune system that is constructed with similar class of cells with this antigen

In this situation, a group of users is entered the system with profiles similar with the test data stored in the database. When the first user enters the system as in the first situation, system will observe its average values of fitnesses, and for second user in the group also the average values are calculated, this will continue like that until the last user enters to the system. At each iteration a single user is permitted to enter into the system. The count of users used in this situation is 100. Like in the first situation the first user enters the system, and an antigen object is created with the fields showing the demographic data, interests and rated locations. Then a TCell object is created, also a TCell population which is constructed from the former user's demographic data, is created by the system. After that, a fitness score is calculated like the one shown in Figure 5.4. The first and the third situations are the same until here, but at that time, the average of the TCell population is calculated, to see the overall performance of the system. The average fitness of TCell population is calculated for the first user and plotted as the fitness of the first TCell in the Figure 5.20. Since every time similar users enter the system, the fitness calculated from the

demographic data of the overall system increases slightly and the average fitness values (Figure 5.20) do not fluctuated like the one in Figure 5.4 since similar users provide reinforcement in fitness values.

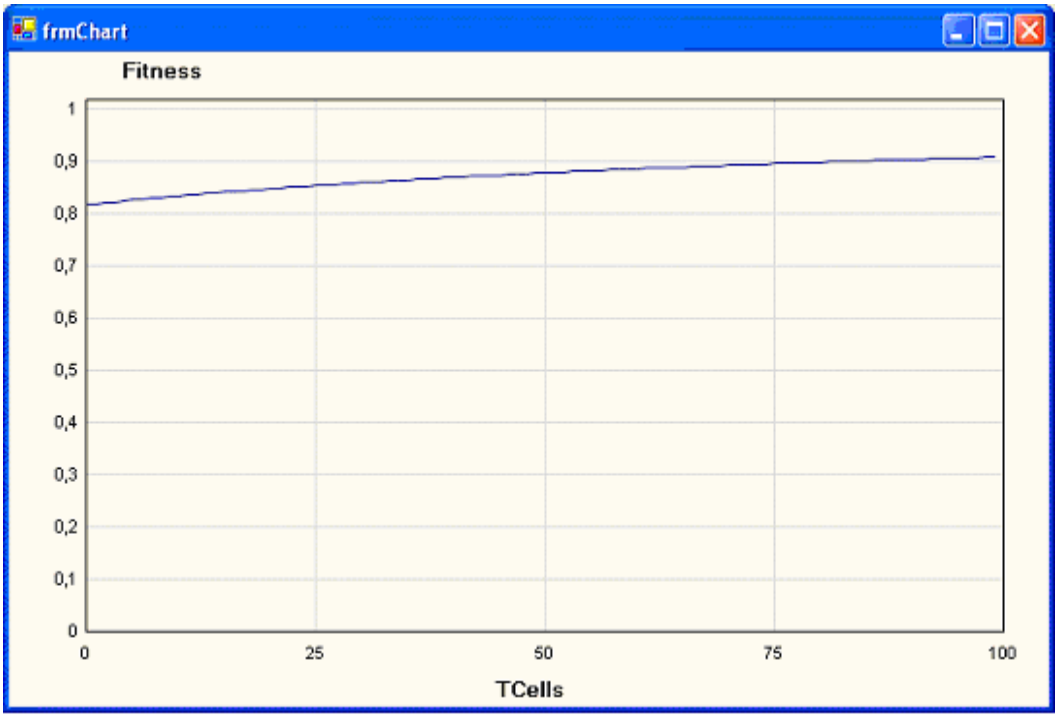

Figure 5.20 TCell population average fitness calculated for each user entered to the system

After calculation of TCell fitnesses, the BCell is created from user's rated items, and the BCell population constructed from places rated by previous users is created for each of the users in the group. Then a similarity measure is calculated for each BCell in the BCell population like in the first situation. Different than the first situation, in the third situation the average similarity of the BCell population is needed and calculated. In Figure 5.21, the horizontal axis shows the IDs of users entering the system at each iteration, and in the vertical axis the average similarity measures of the BCell population are given. Since similarly rated locations are put into the system at each iteration, the average similarity of the BCell population increases. Since the graph is plotted by using average values, the curve in this figure does not fluctuate like the one in Figure 5.5 again due to reinforcement provided by the graph of similar users.

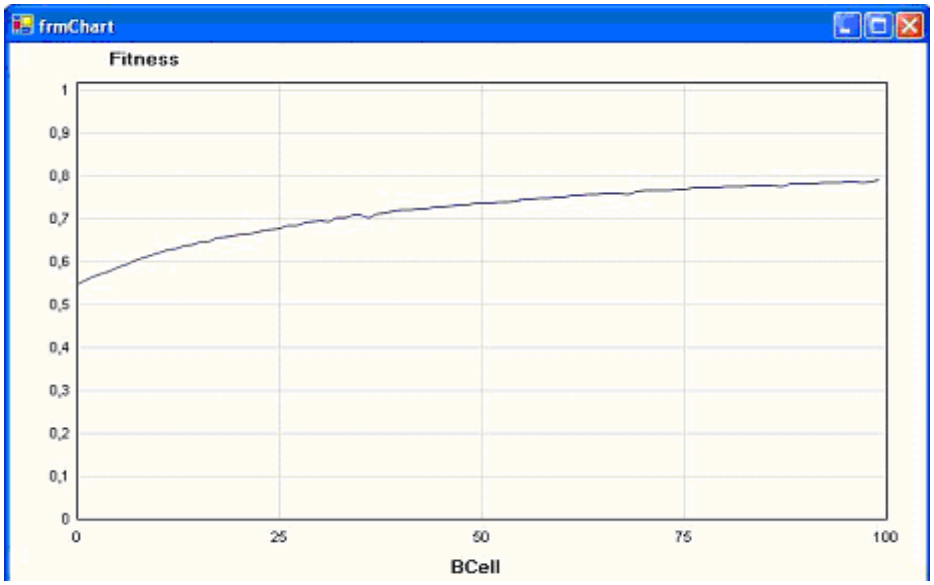

Figure 5.21 Average similarity measure of BCell population calculated for each user entered to the system

After calculation of the similarity measures of BCells, the antibodies are created by using the interests of the people. The average fitness score of antibodies is calculated for each user that entered the system iteratively. Since at each iteration a user with similar interests with the ones in the database is added to the system, the average fitness of the antibodies is increased at each iteration as shown in Figure 5.22.

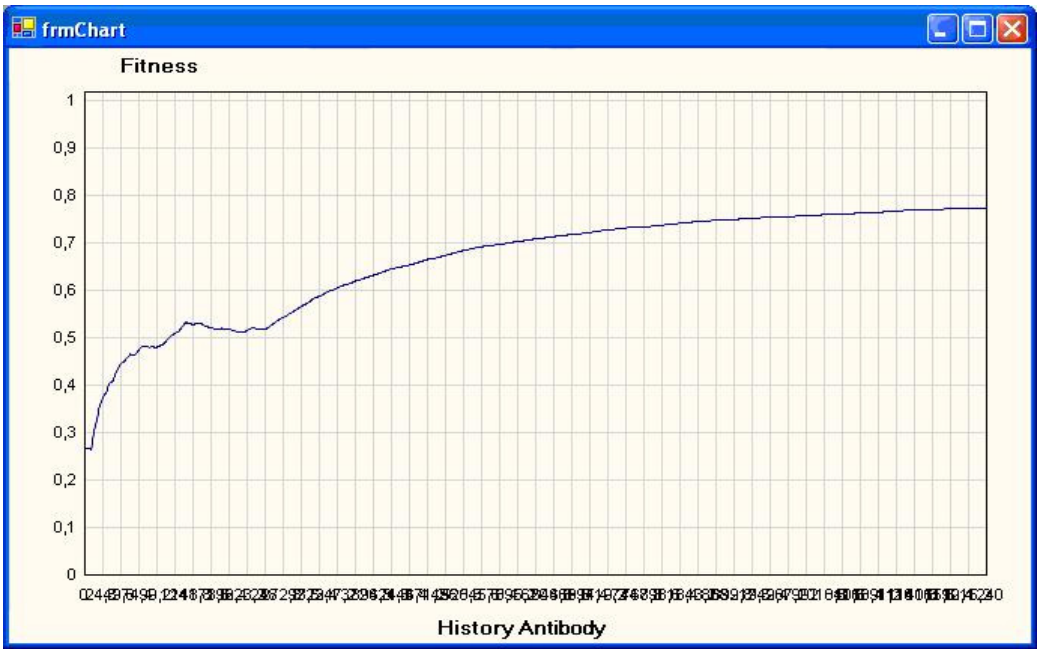

Figure 5.22 Average antibody fitness is calculated for each user entered to the system

After secretion of the antibodies, a subgroup of BCells is selected from the BCell population. Mutation is applied on these BCells. After mutation, the average similarity measure of the BCell Population is calculated. In Figure 5.23, the horizontal axis shows the BCell IDs of users entering into the system, and in the veritcal axis the average similarity measures of the BCell population after mutation is presented for each iteration. In figure 5.9 of the first situation, it was shown that the mutation operator decreased the average similarity measure slightly. In figure 5.23, it can be clearly observed that, despite in some iterations the mutation operator can decrease the average similarity measure, in the overall picture, the mutation operator increases the similarity measure of the BCell population which is clearly observed by a measure that increases from 0.6 to 0.75.

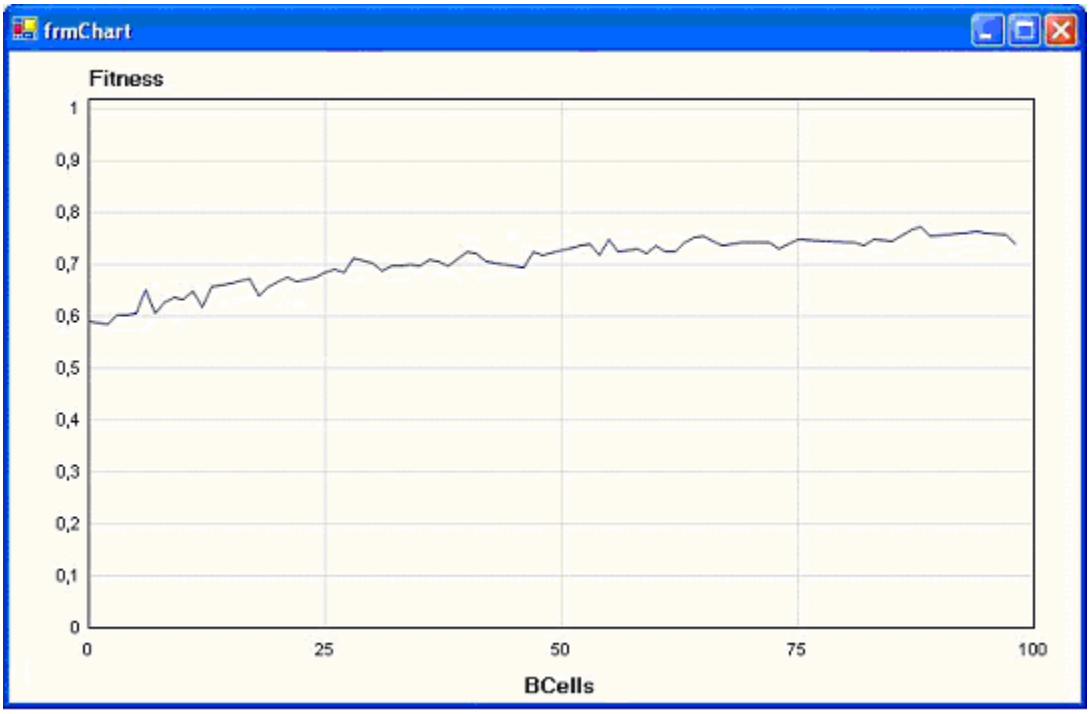

Figure 5.23 Average similarity measures calculated for BCell population after mutation

After mutation this subgroup of BCells is entered into a two stage selection process as in the first situation. Two constants are important in this two stage selection process: the PLACES_BCELL_MIN_FITNESS, which defines the minimum threshold value for BCell similarity measures and is set to 0.6 (Figure 5.2), and the PLACES_BCELL_MAX_FITNESS set to 0.99, which is the maximum threshold

value. By applying this selection algorithm the average fitness of the overall population is increased to about 0.8 (Figure 5.24) from spanning a range of 0.6-0.7 (Figure 5.23). Since in the third situation we process average values, the increase in average similarity measures is more accentuated (Figure 5.23) than for the first situation (Figure 5.10).

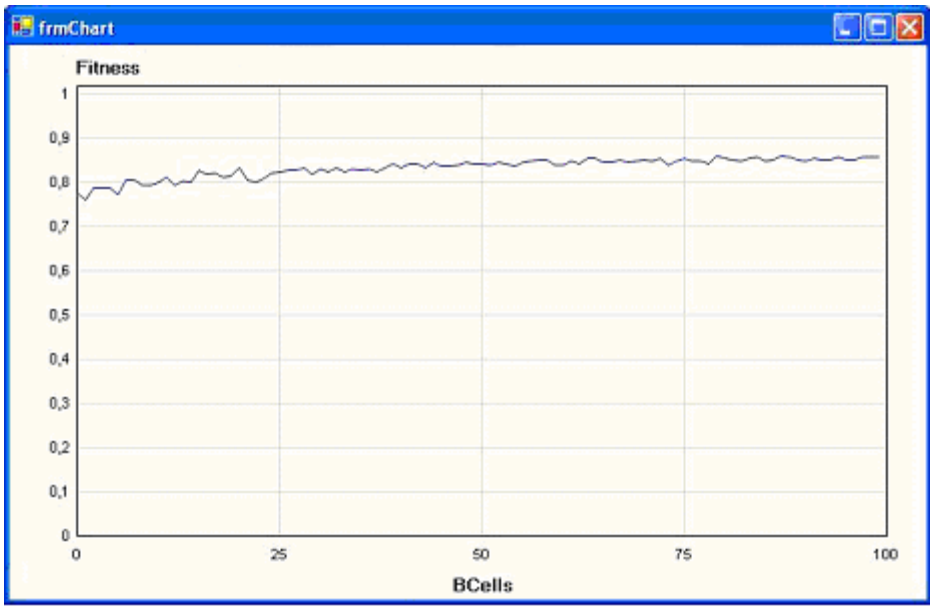

Figure 5.24 Average similarity measures calculated after positive and negative selection

Then finally the average similarity measure of each recommended BCell that belongs to the users entering to the system iteratively is shown in Figure 5.24. The horizontal axis in this figure shows the Ids of users entering the system, and the vertical axis gives the average similarity measure of recommended BCells. The average value for similarity measure spans the region of 0.75 and 0.9, which was spanning the region of 0.85 and 1 in the first situation, because in the first situation decrease in fitness of a recommendation BCell will decrease the average fitness of recommendation BCells that belongs to the user. In the first situation since the average values are taken it can be observed a slight increase in the average fitness of recommendation BCells, which means since new users' profiles are similar, the recommendation BCells resemble to the BCells in the database.

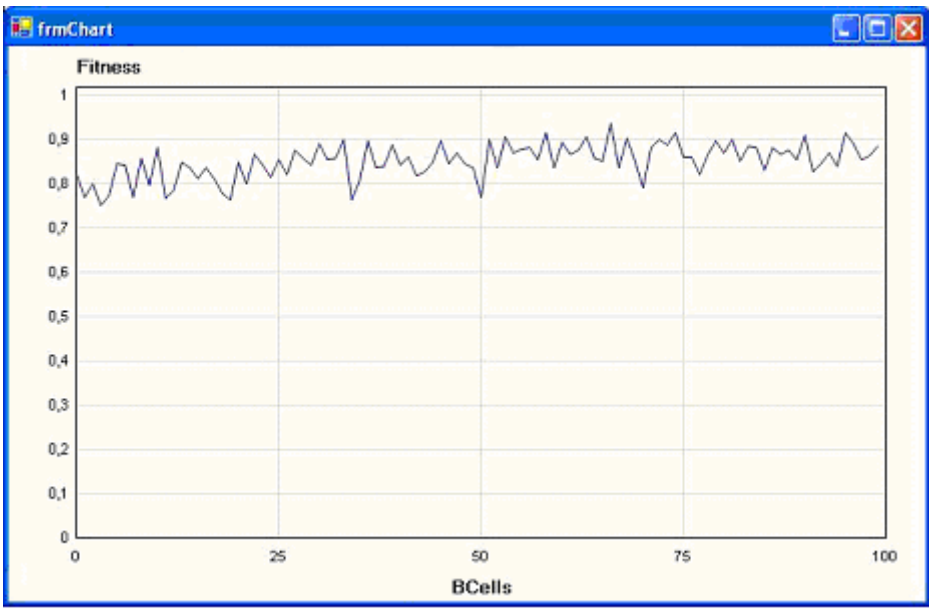

Figure 5.25 Average similarity measures calculated for recommendation BCells

5.5 Test situation 4: A group of antigen put in an immune system that is constructed with different class of cells with this antigen

In the final situation, like in the third situation a group of users are entered into the system, but this time, the new users' profiles are in different class with the user profiles reside in the database. The same database is with the one used in the second situation. The group entering the system possessing the profiles of users that is similar to the German woman in the second situation which has a different profile than the users reside in the database. As in the third situation, at each iteration a single user is permitted to enter into the system and the count of users is 100. Like in all situations, the first user enters the system, and an antigen object is created with the fields showing the demographic data, interests and rated locations. Then a TCell object is created, also a TCell population which is constructed from the former user's demographic data, is created by the system. After that, average fitness of the TCell population is calculated to see the overall performance of the system. As in the third situation, the average fitness of TCell population is calculated for the first user and plotted as the fitness of the first TCell in the Figure 5.26. Since every time similar users enter the system, the fitness calculated from the demographic data of the overall system increases immediately and the average fitness values (Figure 5.26) do not fluctuated like the one in Figure 5.13 since similar users provide reinforcement in

fitness values. Since the new users entering the system has a different profile than that reside in the database, the former values in Figure 5.26 are closer to zero, and since the system learns the new users' profile at each iteration, the average fitness value of the system increases to 0.5. In the third situation, since the profiles of users entering the system is already similar with the ones in the database, the increase in the average fitness of the TCells (Figure 5.20) is softer than the increase in the fourth situation (Figure 5.26).

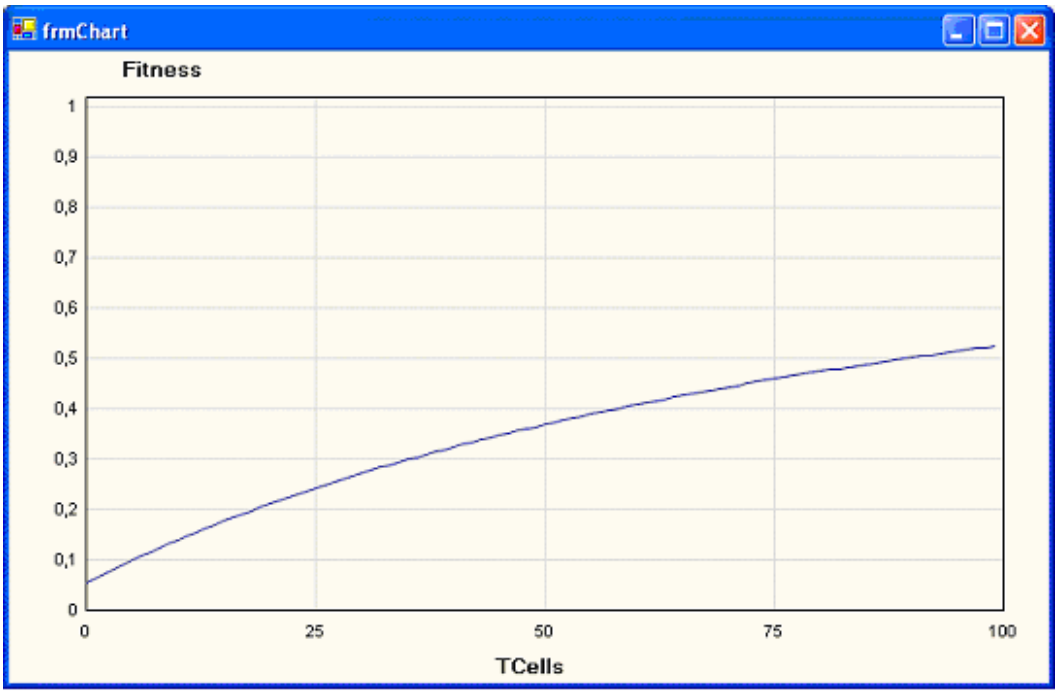

Figure 5.26 TCell population average fitness calculated for each user entered to the system in the fourth situation

After calculation of TCell fitnesses as in the third situation, the BCell is created from user's rated items, and the BCell population constructed from places rated by previous users is created for each of the users in the group. Then a similarity measure is calculated for each BCell in the BCell population like in the third situation. The horizontal axis shows the IDs of users entering the system at each iteration, and in the vertical axis the average similarity measures of the BCell population are given. Since the ratings of users are different from the user ratings in the database, the tangent of the curve in the Figure 5.27 is greater than the one in Figure 5.21, because the system learns the new users profile very fast. It can be easily the observed from Figure 5.27, since the system adapts to changes in the trends, it can be said that

our immune system is highly adaptive

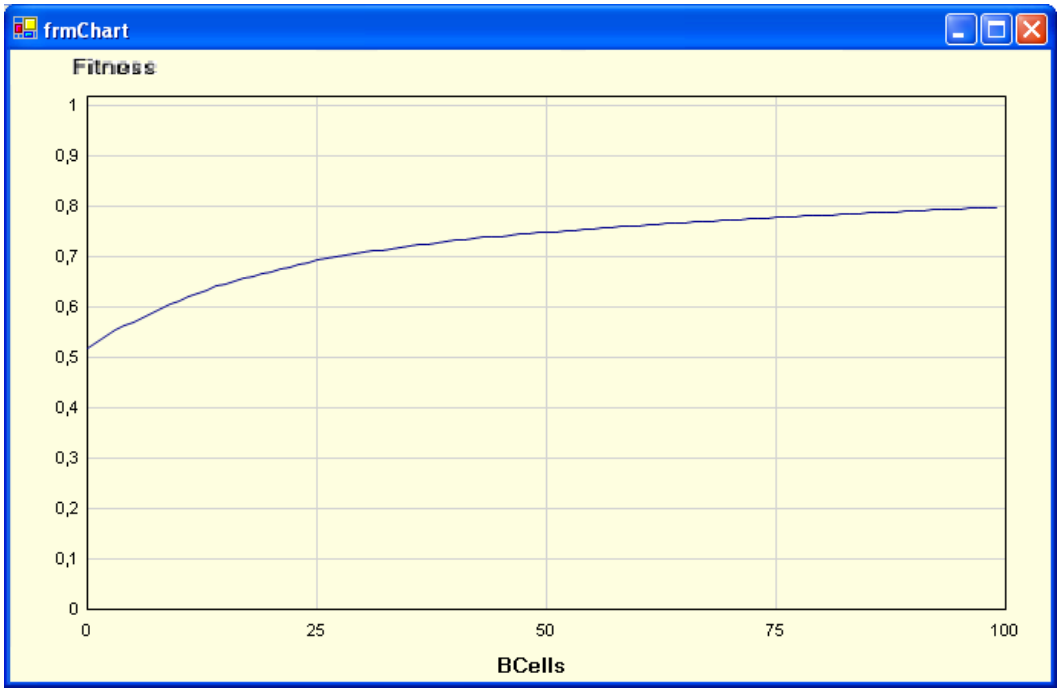

Figure 5.27 Average similarity measure of BCell population calculated for each user entered to the system in the fourth situation.

Like in the third situation, after calculation of the similarity measures of BCells, the antibodies are created by using the interests of the people. The average fitness score of antibodies is calculated for each user that entered the system iteratively. Like in the third situation (Figure 5.22.), the average fitness of the antibodies is increased at each iteration as shown in Figure 5.28. Both Figures shows that our system is highly adaptive and also after learning the interests of new users, the slope of the curve decreases which means the system becomes error tolerant after adaptation, and small changes in the interests of new users does not interrupt the overall performance of the system.

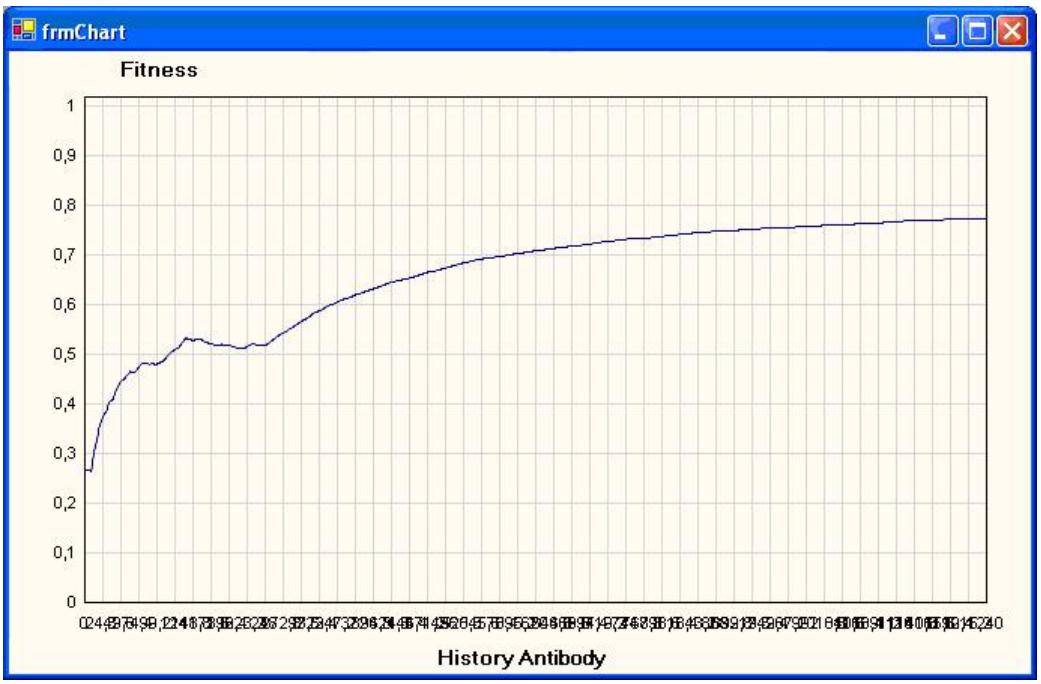

Figure 5.28 Average antibody fitness is calculated for each user entered to the system in the fourth situation

After secretion of the antibodies, a subgroup of BCells is selected from the BCell population as in the third situation. Mutation is applied on these BCells. After mutation, the average similarity measure of the BCell Population is calculated like in the third situation. In Figure 5.29, the horizontal axis shows the BCell IDs of users entering into the system, and in the vertical axis the average similarity measures of the BCell population after mutation is presented for each iteration. Since in Figure 5.14, after mutation the curve spans the whole region between 0 and 1, Figure 5.29 starts with a fitness 0.57 which is smaller than the value which is 0.6 in figure 5.23. But at the end the system learns the new visitors' profiles and the last fitness value in the both figures is similar.

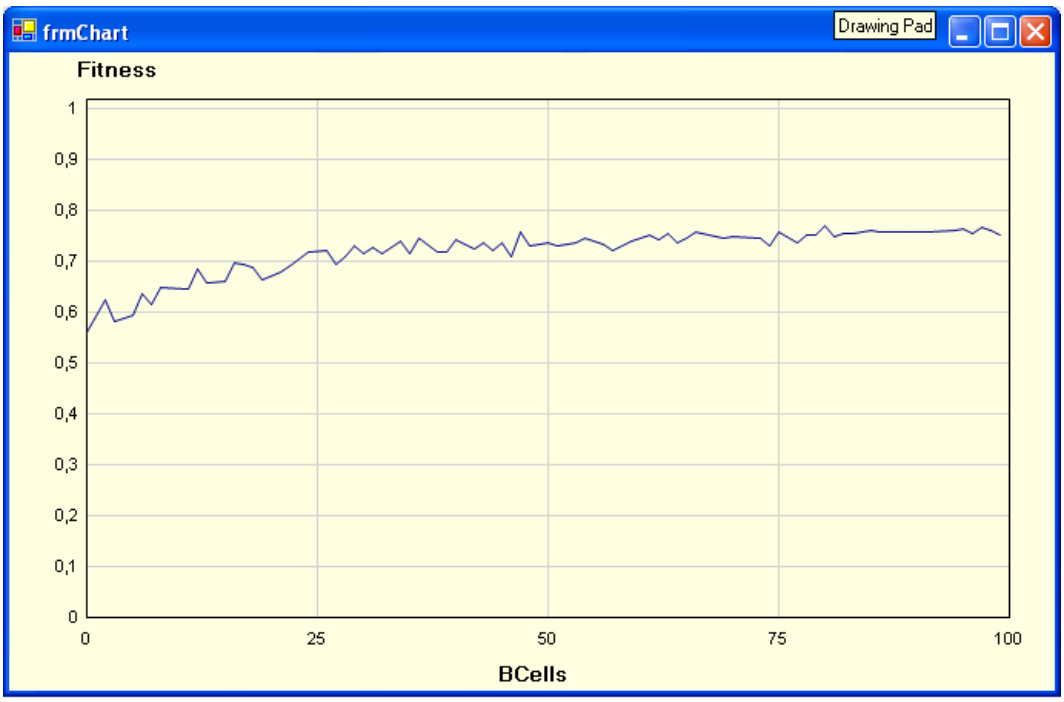

Figure 5.29 Average similarity measures calculated for BCell population after mutation in the fourth situation.

After mutation this subgroup of BCells is entered into a two stage selection process as in the third situation. Two constants are important in this two stage selection process: the PLACES_BCELL_MIN_FITNESS, which defines the minimum threshold value for BCell similarity measures and is set to 0.6 (Figure 5.2), and the PLACES_BCELL_MAX_FITNESS set to 0.99, which is the maximum threshold value. Since Figure 5.24 and Figure 5.30 are similar, we can understand that in situations like the fourth situation, PLACES_BCELL_MIN_FITNESS should be lowered to a value between 0.3 and 0.4, and it should be increased at each iteration for more reliable recommendations. In other words, to increase the system performance, PLACES_BCELL_MIN_FITNESS and PLACES_BCELL_MAX_FITNESS should be changed according to the average fitness of BCells.

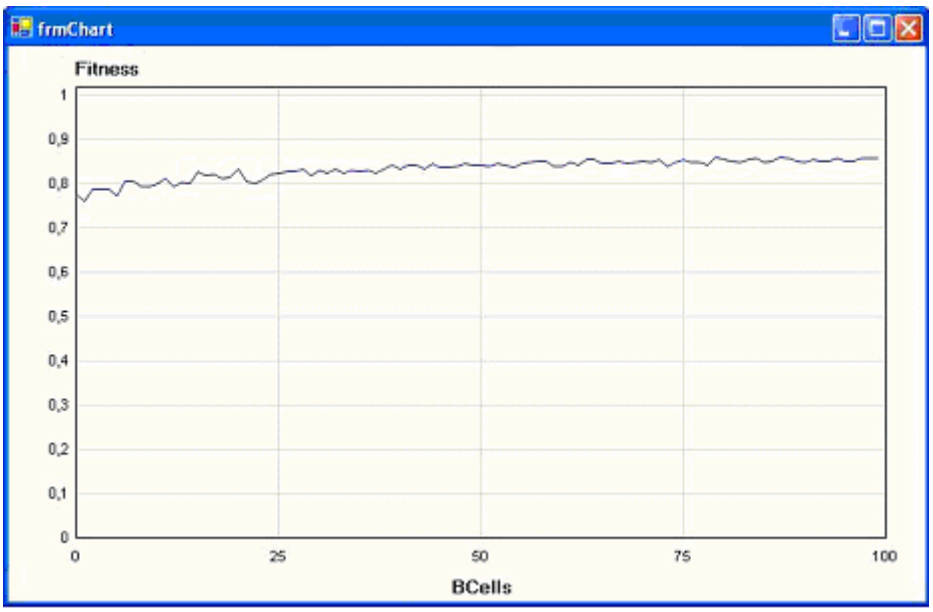

Figure 5.30 Average similarity measures calculated after positive and negative selection for the fourth situation

Then finally the average similarity measure of each recommended BCell that belongs to the users entering to the system is measured as in the third situation. (Figure 5.31). The horizontal axis in this figure shows the Ids of users entering the system, and the vertical axis gives the average similarity measure of recommended BCells. Since the PLACES_BCELL_MIN_FITNESS and PLACES_BCELL_MAX_FITNESS values are not selected properly, in both figures Figure 5.25 and 5.31 the curve spans the same region. Since in the third situation the users are similar, the values PLACES_BCELL_MIN_FITNESS and PLACES_BCELL_MAX_FITNESS are set to a region between 0.6 and 0.99 do not affect the recommendation quality. But in the fourth situation these values should be changed according to the average fitness of BCells.

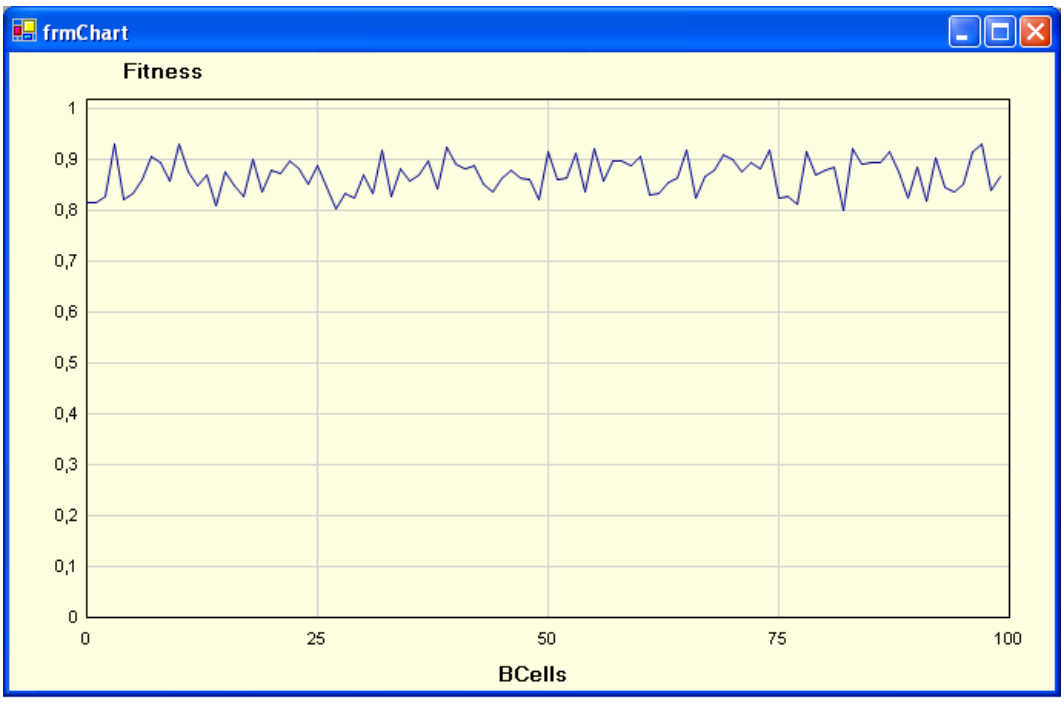

Figure 5.31 Average similarity measures calculated for recommendation BCells in the situation 4

CHAPTER 6

CONCLUSION

6.1 Summary

For achieving to desing an intelligent recommender system that is used as a tourist guide, we have done a sequnce of studies:

In the first chapter, we have stated the objective, and specify the problem. We have argued the properties of the intelligent tourist guide. Next, the motivations that forces us to achieve this goal is determined. Basic architecture of the system is determined, and inputs to the system and outputs of the system are defined. After that in this chapter, the goals that are achieved to realize our goal is stated, and a system flow for such an artificial recommender system is constructed. Then, we have argued that which type of recommender system we have used from two approaches: Collaborative and Case-based approaches. The similarities and differences of these two approaches discussed. In this part of the system, a collaborative filter algorithm that can handle the disadvantages of such an approach, and takes the advantages of case-based algorithms is decided to be used as collaborative recommendation system. Then, general system flow of the intelligent recommender system is designed. Next, the methodologies that can be designed as in a such structure is argued. Artificial immune systems are decided to be fit most in such architecture, and our contributions are presented. Finally in this chapter outline of the thesis is shown.

Then in the secound chapter, surveys that have made during the thesis are presented. This chapter is structured as follows, in the first part recommendation systems are surveyed, and in the second part artificial immune systems are studied in the literature.

In the third chapter, the method which is an artificial immune recommender system is designed. In the first part of the chapter human artificial system is examined. Then

in the second part of the chapter, properties of this system is discussed. Then finally in this chapter our artificial immune system is designed.

In the fourth chapter, the artificial immune system is implemented as a city guide in Eskişehir. The city guide whose name is ESGuide is observed during a scenerio that is formed by traveling a german boy around the city is imagined. The status of the immune cells during exacution of the recommender agent is simulated and demonstrated.

In the final chapter, the behavior of our artificial immune system is observed during four different situations.

6.2 Conclusive Remarks

Our approach has developed an artificial immune system that cancels the disadvantages of collaborative filtering algorithms. Our algorithm is highly adaptive to change in trends, can associate the input to the output by using its associative memory. Since the algorithm uses evolutionary approach, in the output of the system, the system can maintain diversity, and cover the whole solution space, which means it can recommend any item in the database, which is a major disadvantage of a collaborative filter, since the collaborative recommender system different than our artificial immune system can not recommend unrated items.

Our artificial immune system can be used as a pattern recognition system, which can classify a great number of different patterns that are found in different levels of clusters. Immune system uses different types of operators to generate recommendations: positive and negative selection, mutations etc.

In the tests, it is observed that our artificial immune system is highly adaptive, since it can adapt the changes in trends in a small amount of time. The mutation operator used in the algorithm causes the recommendations to cover the whole solution space which means the unrated items can be recommended by our algorithm. It is

understood that the reliability of the system highly depends on the constants PLACES_BCELL_MIN_FITNESS and PLACES_BCELL_MAX_FITNESS.

6.3 Future Works

Since we have worked on the test data in Chapter 5, the results of the tests can be irrevelant with the results that can be observed in the real life. The algorithm should

be tested in a live tourist guide system to observe the behaviors of tourists, and the responses of the artificial immune system.

Another artificial recommender systems, surveyed in Chapter 2 can be used as a banchmark, and the tests in Chapter 5 can be run by using these systems and our artificial recommender system to see the preformance issues.

The recommender system could be run in a huge database that is formed by different classes of user profiles, and performance of the system should be measured in this situation.

REFERENCES

[1] Pemberton, D., Rodden, T., and Procter R., *GroupMark: A WWW Recommender System Combining Collaborative and Information Filtering, 6th ERCIM Workshop,* "User Interfaces for All" CNR-IROE, Florence, Italy, 25-26 October 2000.

[2] Ricci, F., Arslan, B., Mirzadeh, N., and Venturini, A. *ITR: A case-based travel advisory system,* In S. Craw & A. Preece (Eds.), 6th European Conference on Case Based Reasoning (ECCBR), Aberdeen, Scotland, 2002.

[3] Pazzani, M.J., *A Framework for Collaborative, Content-Based and Demographic Filtering*, Department of Information and Computer Science University of California, Irvine, CA 92697, 1999.

[4] Linden, G., Brent, S., and York, J., *Amazon.com Recommendations: Item-to-Item collaborative filtering*, IEEE Internet Computing February, 2003.

[5] Herlocker, L.J., Konstan, A.J., and Riedl, J., *Explaining Collaborative Filtering Recommendations*, Dept. of Computer Science and Engineering University of Minnesota, Minneapolis, USA, 2000.

[6] Smyth, B. and Cotter, P., Ujjin, S. Bentley, and J. Peter, *Content Personalisation for WAP-Enabled Devices*, Computer Science Dept., University College Dublin, Particle Swarm Optimization Recommender System University College London Department of Computer Science Gower Street, London WC1E 6BT, 2004

[7] Burke, R., *Semantic Ratings and Heuristic Similarity for Collaborative Filtering,* Department of Information and Computer Science University of California, Irvine, 2000.

[8] Anderson, M., Ball, M., Boley, H., Greene, S., Nancy Howse Daniel Lemire, *RACOFI: A Rule-Applying Collaborative Filtering System,* Internet Logic Group Institute for Information Technology e-Business National Research Council of Canada Fredericton, NB E3B Sean McGrath, 2003.

[9] Raymond, J.M., Loriene, R., *Content Based Book Recommending Using Learning for Text Categorization,* Appears in Proceedings of the Fifth ACM Conference on Digital Libraries, San Antonio, TX, pp.195-240, June 2000.

[10] Coyle, L. and Cunningham, P., *Exploiting Re-ranking Information in a Case-Based Personal Travel Assistant,* Department of Computer Science Trinity College Dublin, 2003.

[11] Keith B., Rachael, R. and Barry, S., *Case-Based User Profiling for Content Personalisation*, Smart Media Institute, Department of Computer Science University College, Belfield, Dublin, Ireland, 2000.

[12] Dasgupta, D., Nii, Attoh-O, *Immunity-Based Systems: A Survey,* Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics,1997.

[13] Sangüeza, R., VÁZQUEZ-HUERGA, A. and VÁZQUEZ-SALCEDA, J., *Mixing Collaborative and Cognitive Filtering in Multiagent Systems,* Software Departament, Universitat Politècnica de Catalunya Campus Nord, Mòdul C-6, Despatx 204 C/Jordi Girona Salgado, 1-3 08034 Barcelona, 2004.

[14] Balabanovic, M., Shoham Y., *Fab: Content-Based, Collaborative Recommendation,* Communications of the ACM, vol. 40, pp. 66-72, March 1997

[15] Castro, L. Nunes, Fernando, J. Von Zuben, *An Evolutionary Immune Network for Data Clustering*, Proceedings of the VI Brazilian Symposium on Neural Networks (SBRN'00), pp.84, 2000.

[16] Greensmith, J., *New Frontiers For An Artificial Immune System*, Digital Media Systems Laboratory HP Laboratories Bristol HPL-2003-204, October 7th. 2003.
[17] Hunt, J. E. and Cooke, D., *Learning using an artificial immune system*, , Network Comput. Appl., vol. 19, pp. 189-212,1996.

[18] Jerome, H. Carter, *The Immune System as a Model for Pattern Recognition and Classification*, MD University of Alabama—Birmingham, Birmingham, Alabama, 1999.

[19] Roger L. King, Aric B. Lambert, Samuel H. Russ, Donna S. Reese *The Biological Basis of the Immune System as a Model for Intelligent Agents***,** Pages: 156 – 164, 1999.

[20] Twycross, J. and Cayzer, S., *An Immune-based Approach to Document Classification, International Conference on Intelligent Information Processing and Web Mining 2003*, Zakopane, Poland: Springer-Verlag, pp. 33-46, 2003.

[21] Wallace, M., Maglogiannis, I., Karpouzis K., Kormentzas G., and Kollias S., *Intelligent one-stop-shop travel recommendation recommendations using an adaptive neural network and clustring of history*, 2004.

[22] Cayzer S., Aickelin U., *A Recommender System based on the Immune Network*, Proceedings CEC, pp 807-813, 2002.

[23] Forrest S., Javornik B. et al., *Using Genetic Algorithms to Explore Pattern Recognition in the Immune System*, Evolutionary Computation,1992.

[24] Toru Ohira, *Immune Pattern Recognition System,* Sony Computer Science Laboratory Inc. 3-14-13, 26 March 1995.

[25] L.N de Castro and J Timmis, *Artificial Immune Systems: A Novel Approach to Pattern Recognition*, *Artificial Neural Networks in Pattern Recognition*, University of Paisley, pp. 67-84, January 2002.

[26] Cayzer, S., Aickelin, U., *A Recommender System based on the Immune Network*, Proceedings CEC, pp. 807-813, 2002.

[27] Olfa, N., Fabio, G., Cesar, C., Carlos R., and Dipankar D., *A Scalable Artificial Immune System Model for Dynamic Unsupervised Learning,* GECCO-2003 (Conference proceedings), pp. 219-230, 2003.

[28] Watkins A., Timmis J., and Boggess L., *Artificial Immune Recognition System (AIRS): An Immune Inspired Supervised Machine Learning Algorithm,* Genetic Programming and Evolvable Machines, 5(1), March 2004.

[29] MacInnes J., *Learning Dynamic stereotypes for opponent modelling*, In partial fulfilment of an Interdisciplinary PhD, 2001.