Autonomous Agents for Web Pages Filtering

Flávia Coimbra Delicato, Luci Pirmez e Luiz Fernando Rust da Costa Carmo
NCE/UFRJ - Núcleo de Computação Eletrônica – Federal University of Rio de Janeiro
E-mails: flavia@eng.uerj.br, luci@nce.ufrj.br, rust@nce.ufrj.br

ABSTRACT

With the current growth of the information available in Internet, users are facing an information overload. This work proposes a multiagent system for Web pages personalized filtering. The system is composed of a set of autonomous and adaptive agents that automatically provide relevant documents to the user according to a preferences profile. The agents learn with the user feedback and attempt to produce better results over time. This work presents the system description and the promising results of tests performed in a simulated environment. The proposed system proved to be a useful tool to recommend successfully relevant information to a well-defined preferences user.

1 INTRODUCTION

The use of the Internet has been growing in the last years with the appearance of World Wide Web. The exponential increase of computer systems that are interconnected in on-line networks has resulted in a corresponding increase in the amount of information available. Although the increase of the information available facilitates the spreading of knowledge and the acquisition of products and services, it also makes the search for relevant material a real challenge. New tools have been designed with the purpose of locating, filtering and organizing the huge amount of available information.

Recent work that arises at the intersection on information retrieval and software agents offers some new solutions to this problem. Information retrieval is a well-established field of information science that addresses the retrieval from a large set of documents in response to user queries. Agent research is a relatively new field of study, which has grown out of artificial intelligence.

Agents can be defined as softwares with the aim of performing tasks for their users, usually with autonomy, playing the role of personal assistants. Users can delegate to the agents the execution of repetitive and time-consuming tasks.

In order to be of real usefulness for its users, the agents have to learn their interests and habits using techniques of machine learning. They also should be able to adapt themselves to the changes in the users interests, while at the same time they explore new domains of potential interest to the user.

The present work suggests the use of autonomous agents for the personalized information filtering. The proposed system is composed of a set of adaptive and non-mobile agents aiming to satisfy the user's needs for information. The agents receive the user's feedback about the relevance of the retrieved information and improve their search, obtaining better results over time.

The set of agents is autonomous as it can perform its task without the user's presence, based on a preference profile previously built. Besides that, the agents can process information from Internet without keeping the connection during all the time of processing. This is one of the features that classify the agent as autonomous [12].

The system is adaptive as it learns the user's preferences and adapts itself when these ones change over time. The main agent's learning mechanisms is the relevance feedback, widely used in information retrieval systems [14]. The use of genetic algorithms [8] as a complementary mechanism aiming to introduce diversity in the system's parameters is addressed. The information is represented by the vector space model [15], where queries and documents are represented as vectors in a vector space. This method was chosen for its efficiency proved in various works in the area of information retrieval [16][4] and for its relative easy implementation.

The results presented were obtained through a series of sessions with simulated users. The system's efficiency evaluation was made through the normalized distance performance measure (ndpm), suggested by Yao [20].

This paper is organized as follows. In section 2 there's a comparison with related works. An overview of concepts is given in section 3. Section 4 describes the system and the development methodology. The system architecture is detailed in section 5. The analysis of results is presented in section 6 and, finally, some conclusions are drawn in section 7.

2 RELATED WORKS

In the past few years many attempts have been made towards the development of agents that assist in dealing with the huge amount of information available. In the domain of Web , WebWatcher [2] and Lira[3] are agents whose actions are interleaved with the user’s browsing in Netscape. They run on the server-side and require explicit interaction to indicate interest in topics or particular pages. The Remembrance Agent [13] is an autonomous interface agent that reminds the user of relevant files stored on the user’s local disk. MIT Media Laboratory’s Letizia [10] is an autonomous interface agent designed to assist and provide personalization to the user while browsing the WWW by performing a breadth-first search on the links ahead and providing navegation recommendations. Other agent-based systems use some techniques to try to detect patterns in the user's behavior. For instance, InfoScope [7] learns by using systems based on rules that register interesting topics covered in the past. Recommendations of new topics are based on how recent, frequent and spaced these past topics are. The

main disadvantage of such approaches is that they are restricted to recommendations of topics within the domain of user's past interests. In the same way, assisted browsing systems are restricted to the sections of the Web visited by the user, recommending links starting from them. In contrast, the proposed system looks for new domains for information that can be of potential interest for the user. The user probably never saw before the presented topic.

The Newt system [16] is a software agent which adopts relevance feedback and genetic algorithms to provide personalized filtering of Usenets news. The approach differs from the present work in the application domain. Besides this, Newst uses the traditional method of vector-space representation, as described in [15], while in this work different documents representation were tested.

More similar to our work with regards to application domain and representation are the systems built by Balabanovic [5] and Amalthea, proposed by Moukas [11]. Balabanovic proposed a multiagent system that combines both content-based and collaborative techniques applied to the web pages recommendation. That work adopts the vector-space model, relevance feedback as the learning method based and he suggests the use of genetic algorithms as a possible solution for some of the problems found in the content-based filtering. Amalthea is a system that combines the concepts of autonomous agents and artificial life in the creation of an evolving ecosystem composed of competing and cooperating agents. A co-evolution model of information filtering agents that adapt to the various users interests and information discovery agents that monitor and adapt to the various on-line information sources is proposed.

3 BACKGROUND

The purpose of this section is to introduce basic concepts and definitions that lay a basis for the system designs and experiments to follow. Section 3.1 presents the idea of autonomous agents, section 3.2 gives an approach of information filtering systems and section 3.3 describes the relevance feedback technique.

3.1 Autonomous Agents

An autonomous agent is a program that works in parallel with the user. Autonomy says that the agent is, conceptually at least, always running [10]. The agent may discover a condition that might interest the user and autonomously decide to notify him. The agent may remain active based on previous input after the use has given other commands or has even turned the computer off.

The role of an agent as a personal assistant requests the ability to act independently and concurrently to the user. An assistant would not have much practical usefulness if he needed to receive explicit instructions and constant supervision during the whole time of his work execution. On the other hand, with autonomy, he can save the user's time, executing in parallel repetitive tasks while the user drives his attention for another tasks.

The filtering of Web pages is a well-suited domain for autonomous agents. Web users claim for some kind of intelligent help, since the direct-manipulation interface of manually following links in a browser is not enough to prevent them of being overloaded of irrelevant information. [10].

3.2 Information Filtering Systems

The information filtering task involves repeated interactions over multiple sessions with the users having long-term goals. It differs from the information retrieval task, where the users typically have a short-term information need that is satisfied in a single session [16].

Information filtering systems assist users by filtering the data stream and delivering the relevant information to the user. Information preferences greatly vary across users, thus, filtering systems must be highly personalized.

Three different approaches can be identified in the filtering systems literature:

- Systems based on the user's profile [6];
- Systems that perform filtering in a cooperative way, sharing information [18]; and
- Systems that use agents, in which mobility, intelligence and autonomy are fundamental factors [12].

3.3 Relevance Feedback

One of the most important and difficult operations in information retrieval is to generate queries that can succinctly identify relevant documents and reject the irrelevant ones. Users often submit queries containing terms that don't match the ones used to index most of the relevant documents and almost always many unretrieved relevant documents are indexed by different terms from the ones in query. This problem has long been recognized as a major difficulty in information retrieval systems [9].

Since the difficulty in accomplishing a successful search at the first attempt is recognized, it is common to perform iteratively searches and reformulate query statement based on the evaluation of previously retrieved documents. The relevance feedback method is usually adopted for automatically generating improved query formulations [14]. A query can be improved iteratively by using an available query vector (of terms) and adding terms from the relevant documents, while subtracting terms from the irrelevant ones. A single iteration of relevance feedback frequently produces improvements of 40 to 60 % in the search precision [15].

4 SYSTEM DESCRIPTION

The present work proposes the use of agents for the personalized information filtering. The proposed system, named Fenix, is composed by a set of autonomous, adaptive and non-mobile agents, aiming to satisfy the users information needs.

An agent is modeled as a set of individual profiles. As a whole, all the profiles in a population try to satisfy the user's interests and adapt themselves to these interests. One user may have various agents, each of them satisfying his/her needs for information about a certain subject.

The agent is responsible for starting the execution of search and filtering tasks, one for each profile. As the tasks are autonomous, they are sub-agents in Fenix system. Each sub-agent, using different search engines, goes through the web pages looking for documents containing the keywords provided by the user. The set of documents obtained undergoes the filtering process, according to the adopted model. The selected documents are the ones with the higher degree of similarity with the respective profile. These documents are provided for the responsible agent that has to gather the results from all the sub-agents, to classify them according to their potential relevance, presenting them to the user. The user can provide positive or negative feedback for the documents. User feedback has the effect of modifying the profile used to retrieve that document.

4.1 Development Methodology

Fenix system was developed according to the object oriented approach. The programming language adopted was Java, by Sun Microsystems Inc., and the development environment was Jbuilder Standard 3.0, by Borland Corporation, that uses JDK (Java Development Kit) version 1.2. The system was implemented as a Java application to be running locally in the user's machine.

4.2 Fenix Features

A number of attributes can be identified on classifying Fenix as a software agent. These features are discussed below.

Autonomy

Fenix is able to make judgements about the documents relevance without the direct intervention of the user. Besides, when the autonomous mode is on, the system starts new search and filtering tasks automatically, based on the keywords of the user profiles. Concern to the execution environment, a local database is created after the searches and the user hasn't to keep connected to the Internet in the posterior stages of processing. This feature classifies an agent as autonomous in respect to the environment [12].

Temporal Continuity

Fenix has an autonomous mode, in which the system remains active all through the time the user's machine is on. He runs in background until accomplishing its searches for all the user agents. However, the user's machine needs to be on, since the agents reside locally and not in a remote server.

Adaptation Ability

Fenix learns through its learning mechanisms, adapting to the user's interests along the time.

Social Ability

Each user agent starts the execution of several autonomous search and filtering tasks. At the end of all tasks the results are gathered by the controller agent who is responsible for classifying them according to its relevance eliminating repetitions before presenting them to the user. Thus, the Fenix social capacity is established in terms of the interactions among an agent’s tasks, which cooperate to each other aiming a common goal. The interactions are made through messages implemented in the programming language adopted.

Reactivity and Pro-activeness

The system can detect changes in the user's environment as, for example, a document reading and evaluation. In the autonomous mode, if it was detected that all the documents of a profile have already been evaluated by the user, an agent can take the initiative of beginning new searches.

5 ARCHITECTURE

Fenix system is composed of various functional modules (figure1). The modules are implemented as groups of related classes. The description of each module is given below.

5.1 User Interface Module

This module presents a graphic interface to interact with the user. The user's interaction with Fenix system begins with his registration, where he must inform his personal data and choose a login and a password. After the identification the user can choose from three options: to create a new agent, to load an existing one or to activate the autonomous mode.

When creating a new agent, the user must choose a name and a background color, and provide the following search parameters: maximum number of documents shown per session (default is 30); and the query expression. A query in Fenix system is a combination of keywords (technically called terms), separated by blank spaces. The use of logical connectives is not allowed. The presence of the connective AND between the terms is automatically assumed.

As a result of the initial search, a series of retrieved documents is presented. After reading the chosen documents, the user can provide positive (+1) or negative (-1) feedback according to their relevance.

The user can visualize the pages of the documents through the button "Shows". That button activates the local navigator, as, for example, the Netscape or Internet Explorer, that should be configured in the system.

The user can modify a document URL, if he finds a more interesting link starting from the initial page, through the button "Alters". He can also include a URL of interest manually, that has not been retrieved by the agent, by clicking the button “Includes”.

When saving a newly created agent, the references to the documents with positive feedback will be saved (their URLs) and the term vector and their weights will be created, building the initial profiles for that agent.

The user can also choose some URLs to be constantly monitored. Certain URLs are frequently changed and updated, and the system can be scheduled to verify from time to time if their contents changed.

When loading an existing agent, the user can choose one from three actions: to read some retrieved document, to provide feedback about some document or to start a new search for documents.

The Fenix "Autonomous Mode" works performing search and filtering tasks without interaction with the

user. This option is totally controlled by a separate thread, and it works in the following way:

- the system verifies all the agents belonging to the user;
- for each agent, the status of all its documents is verified;
- in case of existing documents not read, the user receives the message informing about that and the corresponding agent's name;
- in case of all the documents of all the agents have already been read, the system initiates new searches for each user's agents; after the searches, a message is presented informing that the user should choose the option of " Shows " for the respective agents.

5.2 Filtering Module

The filtering process consists in translating documents to their vector representations, calculate the similarity between documents and profiles, and selecting the top-scoring documents for presentation to the user.

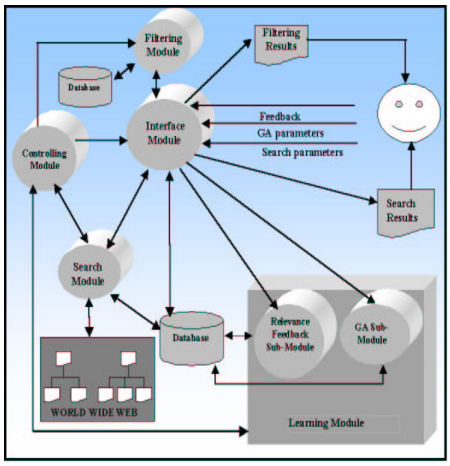

Figure 1: System's architecture

Figure 1: System's architecture

The representation adopted in this work is based on the vector space model (VSM) [15]. According to it, documents and queries are represented as vectors in a hyperspace. A metric distance, which measures the proximity between vectors, is defined over space. The filtering results are the documents with representation that have the highest degree of proximity to the query vector.

A standard method for indexing texts consist in removing punctuation marks, recognizing individual words, eliminating functional words (as "and", "that", etc) using a stop-list, and using the remaining words for content identification of the text. The words may also be truncated, leaving only their stems. This method was adapted for using with web pages. A further step was added, in which the html "tags" were identify, and most of them was removed, except some "meta-tags". "Meta-tags" are commonly used to describe the pages, so they are suppose to be significant for content description purposes.

Since the words (called "terms") are not equally important for content representation, weights are assigned to them, in proportion to their presumed importance to content identification purposes. A text is then represented as a vector of terms and a vector of related weights,

$T_i = \{W_{ij}\}$, where W_{ij} represents the weight of term t_j in text T_i.

In the adopted representation, the term weight is the product of the term frequency and the inverse document frequency. This is a frequently used representation in information retrieval systems [15]. The term frequency (tf) is the occurrence frequency of the term in the text and it usually reflects the relevance of this term. The inverse document frequency (idf) is a factor that enhances the terms that appear in few documents, while it devaluates the terms occurring in many documents. As a result, the documents specific features are highlighted, while the ones spread through the set of documents have minor importance. The weight of the terms is then given as:

$$W_{ij} = tf_{ik} \quad X \quad idf_k, \qquad (1)$$

where tf_{ik} is the number of occurrences of term t_k in document i, and idf_k is the inverse document frequency of term t_k in the collection of documents. A commonly used measure for idf is $idf_k = \log (N/nk)$, where N is the total number of documents in the collection, from which nk contain a term t_k. In this work, a collection of documents is formed by all the documents retrieved by a profile.

A profile is a set of information about the retrieved documents. Such information are, for example, the documents location in the net (URL), the score computed for the system to the documents and the user's feedback assigned to them. Besides, it contains the vector representation of all documents that received positive feedback. The representation consists of a vector of terms similar to the one previously described for documents.

5.2.1 Evaluation of Filtered Documents

A commonly used similarity measure in the vector space model is the cosine of the angle between vectors. Many authors suggest other measures, as we can see in [19]. In the proposed application, different formulae were tested, and it was adopted the one proposed in [15] as follow:

$$S(F_i^d, F_i^p) = \frac{\sum_k w_{ik}^d \, w_{ik}^p}{\sqrt{\sum (w_{ik}^d)^2 \sum (w_{ik}^p)^2}} \qquad (2)$$

where "d" indicates that the field belongs to a document and "p" indicates that the field belongs to the profile.

5.2.2 Scoring and Selecting Documents

The documents retrieved through the search task started by the respective profile will have their similarities calculated in relation to that profile. The agents are responsible for gathering the documents generated by all the profiles, classifying them according to their similarity values, eliminating repetitions and presenting to the user.

The similarity values are converted to a class scale to be presented to the user. This scale seems to be a more natural representation of the document presumed importance than a set of decimal values. A five points scale was adopted, with the adjectives:

- Terrible: for scores equal to 0.2
- Poor: for scores between 0.2 and 0.3
- Neutral: for scores between 0.3 and 0.5
- Good: for scores between 0.5 and 0.8
- Excellent: for scores greater than 0.8

The maximum score value is 1.0 and it only happens when the profile and the document representations are identical. In this work, the limit of 0.2 it was adopted as the minimum score that a document should have to be presented to the user.

The user's feedback for a document has the effect of modifying the respective profile. The profile has to incorporate the changes before new documents can be evaluated.

Since the term vector for a document with feedback is available, the profile is modified according to equation (3) described in the Learning Module below. The existing terms are weighted again, and new terms are included.

5.3 Learning Module

The learning methods addressed in this work were relevance feedback and genetic algorithms. Both methods were designed as independent sub-modules. At the present stage, the relevance feedback sub-module was completely implemented and tested. The specifications of the genetic algorithm sub-module had already been done, but its implementation will be a matter of future works.

5.3.1 Relevance Feedback Sub-Module

In the relevance feedback method, an original query vector (represented by the profiles) is modified based on the user's feedback for the documents retrieved by the profile.

For vector space representations, the method for query reformulation in response to user's feedback is vector adjustment. Since queries and documents are both vectors, the query vector is moved closer to vector representing documents with positive feedback, and further from vectors of the documents with negative feedback.

Take a profile P, which contributed to a document D for presentation to the user. The user provides feedback, which is a positive or negative integer. Each term in the profile is modified in proportion to the feedback received:

$$\forall i, k: \; W_{ik_p} = W_{ik_p} + \alpha * f * W_{ik_d} \qquad (3)$$

that is, the weight of each term is changed proportionally to the learning rate (α) and to the feedback. The learning rate α indicates the sensibility of the profile to the user's feedback, and, in general, assumes values between 0.5 and 1 [16].

The effect is that, for those terms already existing in the profile, the term weights are modified in proportion to the feedback. The terms not existing in the profile must be added to it.

Relevance feedback is not the only way the user can alter the dinamycs of the system. The user can explicitly introduce new URLs or links he considers interesting, by clicking the “Alter” and “Include” interface buttons.

5.3.2 Genetic Algorithm Sub-Module

In the next stage of this work the genetic algorithm will be implemented as a complementary mechanism to introduce diversity to the search parameters as a goal. This goal will be achieved by recombining the contents of different vectors of terms belonging to the same user profile. Genetic operators, like mutation and crossover will be applied.

The formal definition of a population P in an iteration t is given in the equation below. P is defined as a group, where each element is a pair of profile and its fittness:

$$P(t) = \{(p, f(p))\} \qquad (4),$$

where p represents a profile and f (p) its fittness. Each profile is converted for a binary representation, and it corresponds to an individual or chromosome of the population. The fittness is computed based on the average values of similarity between the documents and their respective profiles. The genetic operators of crossover and mutation update the population to each generation, introducing new members and taking advantage of the fittest ones. The final objective is to evolve the population in direction to a global optimization.

The use of genetic algorithms can possibly prevent a frequently reported problem in information retrieval works, the "over-fitting" [5]. When a profile become too much specialized in his users interests, the filtering results degraded. It occurs because the profile doesn't assign high scores except to the documents that exactly matches him. In this context, the genetic algorithm would be able to revitalize the profile content and to prevent the stagnation.

5.4 Controlling Module

This module is composed of three main controlling classes, from which all the other classes are created and their methods are called. ***Principal*** class controls the system's global behavior. It is responsible for creating the instances of the other controlling classes and for coordinating the communication among them. It's also responsible for calling the interface classes that constructs the initial window of the system.

Agent class controls the behavior of each user's set of agents. It identifies the need of an agent creation and calls the methods of class ***PerfilAgente***. It also receives and agregates the search results of different agents, eliminating redundancies and formatting them to present to the user.

PerfilAgente class controls the specific behavior of each agent. It creates one agent and starts its execution, maintains its persistent data by accessing the local database and updates its contents when it is necessary.

Furthermore, there are two classes considered as parts of the controlling module: ***threadauto***, responsible for the autonomous behavior of the system; and ***threadBusca***, that controls the search module execution.

5.5 Other Modules

The search module is responsible for gathering information from web pages about the chosen subject and saving them in a local database. With the development of this work, there was a choice for using existing search engines, such as Altavista[1], Lycos [2]and others. This module became responsible only for making the interface with these mechanisms, providing the user's keywords to them, retrieving the search results and storing the retrieved pages in a local database.

The system database is composed of all information from the user, his agents and respective profiles, as well as the pages retrieved in searches. This database is implemented through a group of classes existent in Java standard APIs. Users and agents data are stored in simple text files. Profiles and documents are stored in object files.

6 RESULTS

In this work, it was adopted the performance measure proposed by Yao [20]. The ndpm measure ("normalized distance-based performance measure") is a distance, normalized to range from 0 to 1, between the user's classification for a set of documents and the system's classification for the same documents. This will provide a relative measure, that will be more appropriate to the system's goal than recall and precision [17] measures, commonly used in information retrieval.

An outline was adopted as suggested in [4]. A special list of documents is supplied to a simulated user who should classify it in agreement with his interests by a subject. That list is randomly selected from several documents retrieved from the web. The system also ranks the documents according to how well they match the profile previously built for that user.

An ordinal scale is adopted to obtain user rankings. The user places each document into one of the five categories: Excellent-Good-Neutral-Poor-Terrible. The scores computed by the system are converted into each category according to a range of values. The expected result is for the ndpm distance between the user and system classifications to decrease gradually over time, as the user's profile is adjusted.

One hundred and twenty agents were created for thirty subjects of interest to a simulated user. For each subject, a certain number of simulated sessions of "user"-system interaction were accomplished. After an initial search, the agents classified the retrieved documents according to the categories above, the "user" evaluated the documents, providing their feedback values and classification. With the feedback, the agents profiles were adjusted to further searches and the classification is used to computer the ndpm.

A progressive decrease of the ndpm distance along the sessions (figure 2) was observed, indicating that the agents were adapting themselves to the user's preferences and increasing the probability of retrieving a larger number of relevant documents while discarding the irrelevant ones.

Several system configuration parameters were tested in the simulated sessions.

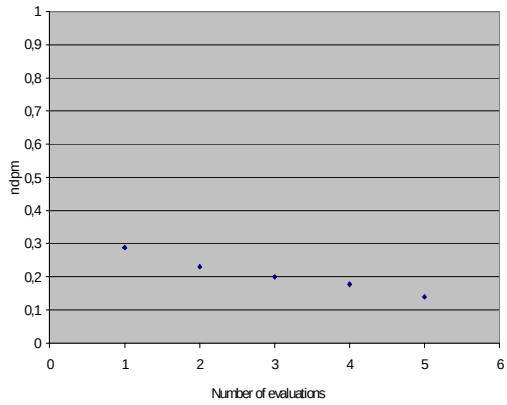

Figure 2: Average ndpm distance between user and system rankings, over all agents at evaluation points.

To sum up the final results we can say that the system reached the best performance when:

- the terms of the query were more specific, in opposite to generic queries;
- the agents were composed by at least 4 (four) and in the maximum 10 (ten) profiles; and
- the term vectors of the documents had maximum size of 300 terms.

Nine agents were tested along 20 (twenty) sessions, in order to compare with the work described in [5],

[1] Available in: http://www.altavista.digital.com
[2] available in: http://www.lycos.com

where a multiagent system was implemented for the WWW pages recommendation. Balabanovic proposed an architecture that combined content-based with collaborative filtering. His system performance was also evaluated with the ndpm measure and the obtained curve had a behavior quite similar to the one presented in the tests with Fenix (Figure 3). In his work 25 evaluation sessions were accomplished. The initial values of ndpm average were of 0.4 and the final values were 0.001.

A question that has arisen in the tests was the low scalability of the system. The number of users and profiles is potentially increasing. As a consequence, the efficiency issue must be addressed. Index structures and algorithms have to be adopted aiming to optimize the profiles and documents processing in order to the system provides the relevant information in a timely fashion.

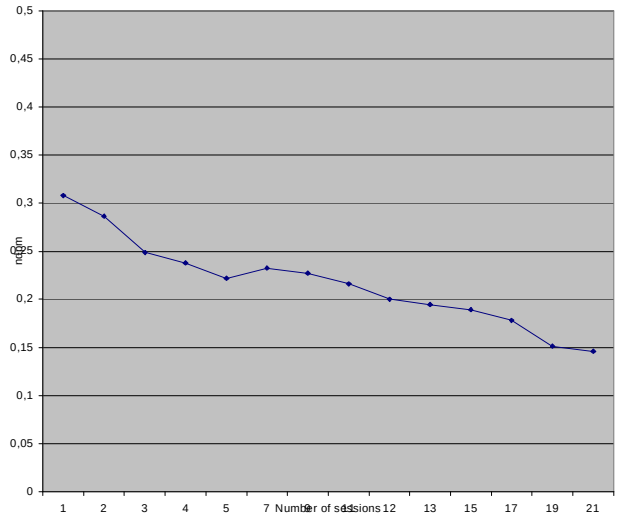

Figure 3: Average ndpm distance between user and system rankings, over 20 sessions.

7 CONCLUSIONS

Fenix is an autonomous agent that must be able to specialize to user interests, to adapt when they change and to explore the domain for potentially relevant information.

The system proved to be a powerful tool of information filtering. The presented results confirmed that the system, using the vector-space model with relevance feedback as the learning mechanism is able to successfully filter relevant documents for a well-defined preferences user.

One important contribution of this work was the adaptation of techniques commonly used for mail and usenet articles filtering to web pages filtering.

The performance values obtained in simulated tests based on the ndpm measure were similar to the ones found in another works of information filtering. Balabanovic [5] proposed an approach that combines content-based and collaborative techniques applied to the WWW pages recommendation. The results described in his work were quite similar to the obtained with Fenix, in which a simpler approach was adopted.

A problem detected during the tests was the low scalability of the system. As a future work, a possible solution for that problem is the implementation of a system with multiple search agents collecting pages web in several sources. The search agents would deposit the pages obtained in a central repository where individual users' personal agents would recover those that best satisfied the its users' profiles. The supplied feedback would go both to the personal agent (that would adjust the user's profile) and to the search agents, that would be serving the users' groups instead of individual users and taking advantage of the shared interests.

Another subject for future works is the implementation of the genetic algorithm sub-module, aiming to optimize the results of the filtering process.

The impact of systems like Fenix on an enterprise may be quite significant. Several information search tasks could be automated, as the system is able to filter and classify different interesting subjects, so reducing the time consumption and the money spent on those activities.

Information filtering agents are a great promise to the management of extensive available information.

REFERENCES

1. Ackley, D and Littman, M, Interactions between Learning and Evolution. Artificial Life II, v X, pp. 487-509. Edited by C. Langton, C. Taylor, J. Farmer and S. Rasmussen, Addison Wesley, 1992.
2. Armstrong, R. et all., WebWatcher: A Learning Apprentice for the World Wide Web, in AAAI Spring Symposium on Information Gathering, Stanford, CA, March 1995. Available in: http://www.cs.cmu.edu/afs/cs/project/theo-6/web-agent/www/project-home.html.
3. Balabanovic, M. and Shoham, Y., Learning Information Retrieval Agents: Experiments with Automated Web Browsing, , in AAAI Spring Symposium on Information Gathering, Stanford, CA, March 1995. Available in: http://flamingo.stanford.edu/ users/ marko/ bio.html.
4. Balabanovic, M. An Adaptive Web Page Recommendation Service. Stanford Universal Digital Libraries Project Working Papers SIDL - WP. 1997.
5. Balabanovic , M. Learning to Surf: Multiagent Systems for Adaptive Web Page Recomendation Service. Dissertation submitted to the Department of Computer Science and the Committee on Graduate Studies of Stanford University. UMI Number: 9837173. UMI Company. 1998.
6. Brusilovsky, P. Adaptive Hypermedia: an attempt to analise and generalize.1994. Available in: http://www.cs.bgsu.edu/hypertext.adaptive/ um94.html
7. Fischer, G., Stevens, C., Information access in complex, poorly structured information spaces.

Human Factors in Computing Systems CHI'91 Conference Proceedings, 1991, pp. 63-70.
8. Goldberg, D. E. Genetic and Evolutionary Algorithms come of age. Communications of the ACM, 37(3):113-119, March 1994.
9. Lancaster, F. W. 1969. "MEDLARS: Report on the Evaluation of Its Operating Efficiency." American Documentation, 20 (2) 119-48.
10. Lieberman, H., Letizia, an agent that assists web browsing. In *Proceedings of IJCAI-95*. AAAI Press, 1995.
11. Moukas, A., *Amalthaea:* Information Discovery and Filtering using a Multiagent Evolving Ecosystem. In proceedings of the Conference on Practical Applications of Agents and Multiagent Technology, London, April 1996
12. Nissen, M. et al.Intelligent Agents: a Technology and Business Aplication Analysis, BA248D: Telecomunications and Distributed Processing, Intelligencia, Inc, november1995.
13. Rhodes, B.J. and Starner, T., The Remembrance Agent, AAAI Symposium on Acquisition, Learning na Demonstration, Stanford, CA, 1996. Available in: http://www.media.mit.edu/~rhodes /remembrance.html
14. Rocchio, J.J. Relevance feedback in information retrieval**.** In: The Smart Retrieval System - Experiments in automatic Document Processing, p. 313-323, Englewood Cliffs: Prentice-Hall, 1971.
15. Salton, G., Automatic Text Processing – The Transformation, Analysis and Retrieval of Information by Computer. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.
16. Sheth, Beerud. NEWT: A learning approach to personalized information filtering.Thesis.[s.l.:1994]. Available in: http://agents.www.media.mit.edu/groups/agents/ papers/newt-thesis/ tableofcontents2_1.html.
17. Silva, E. B. BSETI - Uma Ferramenta de Auxílio à Busca e Recuperação de Documentos - 1996. Available in: http://www.cos.ufrj.br/~bezerra/pf/PF.html
18. Twidale, M. B., Nichols, D. M. and Paice, C. D.. Browsing is a Collaborative Process.Tecnical Report - CSEG/1/96-Computing Department, Lancaster University, 1996.
19. van Rijsbergen, C. J. Information Retrieval, Butterworths, London, Second Edition, 1979.
20. Yao, Y. Y. 1995. Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information Science 46(2):133-145.