

Learning Fuzzy Models of User Interests in a Semantic Information Retrieval System

Mauro Dragoni and Célia da Costa Pereira and Andrea G.B. Tettamanzi¹

Abstract. We propose an approach to user model-based information retrieval which uses an evolutionary algorithm to learn fuzzy models of user interests and to dynamically track their changes as the user interacts with the system. The system is ontology-based, in the sense that it considers concepts behind terms instead of simple terms.

The approach has been implemented in a real-world prototype newsfeed aggregator with search facilities called IFeed. Experimental results show that our system learns user models effectively. This is proved by both the convergence of the interest degrees contained in the user models population and the increase of the users' activities on the set of proposed documents.

1 Introduction

An Information Retrieval (IR) system aims at retrieving, from a collection of documents, those which are perceived by the user as containing information of value with respect to his personal information need. There are some problems which, if solved, could lead to a real improvement of the set of retrieved documents in terms of information value. For example, conventional IR systems do not consider that users are not always able to specify correctly their information need; that user experiences, goals, knowledge, etc., may significantly affect their judgment in defining a document as relevant or not and that the user's information need may vary over time, for example as a result of interacting with the system.

The earliest approaches to IR claimed that "IR systems should not attempt to understand the content of a document" [12]. Systems constructed upon this hypothesis use a keyword approach to individuate documents which are pertinent with respect to the keywords representing the user query. However, as stated by Giger [6] and later by others, these approaches do not take into consideration that the users invariably associate a meaning with each keyword in order to express their information need. Another inconvenient with keyword-based approaches is that they do not take into account the fact that there are terms which are syntactically independent but which can have the same meaning (synonyms). Ontology-based information retrieval approaches [1, 14] promise to increase the quality of responses since they aim at capturing within computer systems some part of the semantics of documents, leading to more effective information

retrieval systems. This paper presents a User-Modeling-based IR system which attempts to solve the above-mentioned problems with the result of improving the document's information value for the user and with the advantage of increasing the precision of query results. The system considers a set of parameters from the user environment to learn the user's degree of interest toward different sets of document.

The original contribution of the work described is a synthesis of two research directions that are being actively pursued in the field of information retrieval:

1. the use of machine-learning techniques;
2. automated user model acquisition.

The paper is organized as follows: Section 2 begins with an overview of proposed IR methods that use User Modeling, Machine Learning, and Ontologies; Section 3 describes the evolutionary algorithm used for evolving models and implemented in our system; Section 4 presents the experiments that we have carried out and discusses their results. Section 5 concludes.

2 Related Work

Many approaches for improving the effectiveness of IR systems have been proposed in the past years. Among them, we can distinguish those which determine term importance by exploiting concept-based information found in ontologies; those which construct an user profiling thanks to the interaction between the user and the system; and those which predict users' interests by using learning algorithms. In this section, a description of some of such approaches proposed in the literature is provided.

Shen and colleagues [10] introduce a domain-specific concept information retrieval system to facilitate the retrieval of organizational internal documents like technical reports, professional references and even e-mails. The domain knowledge of the proposed system is organized using taxonomy for ease of knowledge storage and sharing.

Baziz [1] introduced an approach to document content representation by ontology-document matching. His approach detects concepts in a document by means of a general-purpose ontology, namely WordNet, and uses two criteria to calculate the weight of a concept: co-occurrence of concepts in a document, and their semantic similarity to compute semantic relatedness and then disambiguate them. The result is the set of scored concepts-senses with weighted links called *semantic core of document* which best represents the semantic content

¹ Università degli Studi di Milano, Dipartimento di Tecnologie dell'Informazione, Via Bramante 65, I-26013 Crema (CR), Italy, email: mauro.dragoni,celia.pereira,andrea.tettamanzi@unimi.it

of the document. A survey of concept-based approaches in information retrieval has been carried out by Haav and Lubi [8].

User modeling can be defined as the process of constructing, maintaining and using user models. The development of user models offers the possibility of individuating users' interests by tracking their information-seeking behavior and evolving information needs over time.

Crabtree and colleagues, [2] derive user interest profiles automatically by monitoring user web and email habits. A clustering algorithm is employed to identify interests, which are then clustered together to form interest themes.

Soltysiak and colleagues [11] describe experimental work conducted to investigate user profiling within a framework for personal agents. In particular, the authors aimed at discovering whether user interests could be automatically classified through the use of several heuristics.

Kelly and colleagues [9] propose a user modeling system for personalized interaction and tailored retrieval that (1) tracks interactions over time, (2) represents multiple information needs, both short and long term, (3) allows for changes in information needs over time, (4) acquires and updates the user model automatically, without explicit assistance from the user, and (5) accounts for contextual factors such as topic familiarity and endurance of need.

Goldman and colleagues [7] propose a system that carries out highly effective searches over collections. The system is comprised of two major parts. The first part consists of an agent, MUSAG, that learns to relate concepts that are semantically "similar" to one another. In other words, this agent dynamically builds a dictionary of expressions for a given concept. The second part consists of another agent, SAg, who is responsible for retrieving documents, given a set of keywords with relative weights. This retrieval makes use of the dictionary learned by MUSAG, in the sense that the documents to be retrieved for a query are semantically related to the concept given.

3 Evolutionary Algorithm

Evolutionary algorithms (EAs) [3, 5] are a broad class of stochastic optimization algorithms, inspired by biology and in particular by those biological processes that allow populations of organisms to adapt to their surrounding environment: genetic inheritance and survival of the fittest.

An EA maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a (usually quite small) set of stochastic operators, known as *mutation*, *recombination*, and *selection*.

Mutation randomly perturbs a candidate solution; recombination decomposes two distinct solutions and then randomly mixes their parts to form novel solutions; and selection replicates the most successful solutions found in a population at a rate proportional to their relative quality.

The initial population may be either a random sample of the solution space or may be seeded with solutions found by simple local search procedures, if these are available.

The resulting process tends to find, given enough time, globally optimal solutions to the problem much in the same way as in nature populations of organisms tend to adapt to their surrounding environment.

In the proposed system, an individual of the population is a

vector of as many components as the document clusters in the repository; each component is the fuzzy degree to which the user is interested in the relevant cluster (0 = not interested at all, 1 = absolutely interested).

3.1 Fitness Function

By definition, the quality of a model should be measured as the degree to which the relevant user is satisfied, overall, with the answers to his queries. However, such overall user satisfaction cannot be measured in a single interaction. The only thing that can be indirectly estimated is user satisfaction with respect to an actual query. Therefore, the fitness of a user model can only be estimated incrementally, by subsequent refinements, every time an answered filtered by it is given feedback by the user. At each moment, the fitness of a user model is known with imprecision, which, by its nature, may be represented as a fuzzy interval [13] in the interval $[0, 1]$, where 0 represents complete user dissatisfaction and 1 indicates complete user satisfaction, which would occur if all queries were answered with the desired set of documents ordered by decreasing user interest.

This fuzzy interval can be represented by means of a trapezoid (a, b, c, d) where (a, d) is the support interval and (b, c) the core interval.

When a random user model is created, there is complete ignorance on its fitness; this is represented by a fuzzy fitness of $(0, 0, 1, 1)$, i.e., the whole $[0, 1]$ interval.

Every time a user submits a query, a model is selected from the relevant population and used to filter the answer; then an implicit feedback is calculated based on the actions the user takes: these are reading a document and spending time on a document. The fuzzy fitness of the model is then updated based on two coefficients:

$r \in [0, 1]$: the rating (implicitly) given by the user (0 = completely negative, 1 = completely positive);

$w \in [0, 1]$: the confidence of the rating.

The updated trapezoid is calculated as follows:

$$a' = (a + wr)/(1 + w); \quad (1)$$

$$b' = (b + wr)/(1 + w); \quad (2)$$

$$c' = (c + wr)/(1 + w); \quad (3)$$

$$d' = (d + wr)/(1 + w). \quad (4)$$

A positive rating moves the trapezoid to the right, a negative rating to the left, and both shrink it, i.e., make it less uncertain. In addition, also the recombination and mutation operators, as explained below, change the fuzzy interval of fitness.

3.2 Recombination

The recombination operator is implemented by means of uniform crossover, whereby the new individual is created by selecting, for each cluster interest, the corresponding interest from either parent with $\frac{1}{2}$ probability.

Following recombination, the fitness of the offspring is calculated from those of the two parents (a_1, b_1, c_1, d_1) and

(a_2, b_2, c_2, d_2) as follows:

$$a = b - (b_1 + b_2 - a_1 - a_2)/2; \quad (5)$$

$$b = \min\{0.95b_1, 0.95b_2\}; \quad (6)$$

$$c = \max\{1.05c_1, 1.05c_2\}; \quad (7)$$

$$d = c + (d_1 + d_2 - c_1 - c_2)/2. \quad (8)$$

The rationale behind these formulas is that the fitness of the offspring can be anything in between the fitness of the parents, or even slightly better or worse. The 1.05 and 0.95 factors represent a 5% expansion of the core of the fitness of the parents, which models a slight decrease of precision. The actual value of these factors is not critical, as previous experiments revealed.

3.3 Mutation

The mutation operator perturbs every cluster interest in a user model. In addition, to speed up convergence, an *intelligent* mutation has been used, that has already been applied with success to a similar problem [4]. The main idea behind intelligent mutation is to inject single, larger perturbations and observe the improvement they bring about: if the fitness increases, the perturbation is maintained, otherwise it is discarded.

In this case, intelligent mutation uses user feedback directly to modify the interest of the cluster whose centroid is closest to the query results.

Following mutation, the trapezoid of fitness is updated as follows:

$$a' = \max\{0, b' - (b - a)(1 + s)\}; \quad (9)$$

$$b' = b(1 - s); \quad (10)$$

$$c' = c(1 + s); \quad (11)$$

$$d' = \min\{1, c' + (d - c)(1 + s)\}, \quad (12)$$

where $0 \leq s < 1$ is the strength of the perturbations undergone by the mutated user model. The idea is that the fitness of a mutated model is more uncertain than the fitness of the original model.

3.4 Selection

Probabilistic tournament selection is used to select individuals for reproduction. A number of individuals are picked at random to play a tournament. A deviate from a probability distribution obtained by normalizing the fuzzy fitness interval of each individual is extracted and the individual whose deviate is greatest wins the tournament and is inserted in the population for the next generation.

Elitism is implemented: the best individual in the population is passed on unchanged to the next generation. Another slot in the new population is taken by the last individual used for filtering a query answer; the rationale for this choice is that the fitness of that individual incorporates the most recent information on user interests.

4 Experiments and Results

To validate the proposed approach, we have implemented a small but working real-world prototype newsfeed aggrega-

tor with search facilities, called IFeed². IFeed is a web-based newsfeed aggregator that is able to crawl news from RSS feeds, to monitor the user's activities (such as queries, clicks, etc.) in order to estimate implicit feedbacks, and to learn user profiles without requesting explicit interactions from users. IFeed is composed of the modules briefly described below.

1. Crawler: this module fetches RSS feeds from the web, extracts news from the XML file containing them, translates them into a conceptual document representation, and stores the news in the database. Documents are represented by means of a conceptual representation which is out of the scope of this paper.
2. User Profile Reader (UPR): when a user logs into the system, the UPR loads the models associated with the logged user, selects one of such models, composes a set of 20 documents that are most interesting with respect to the selected model, and proposes them to the user.
3. User Profile Analyzer (UPA): this module updates the user models based on the implicit feedbacks stored in the database. The UPA implements the evolutionary algorithm explained in Section 3.
4. Web Interface: this is the interface used by the users to interact with the system through their browser.

IFeed is implemented by using two different programming languages: the crawler, the UPR, and the web interface have been implemented in PHP, while the UPA has been developed in Java. Documents and other data are kept in a MySQL database.

The experiments have focused on three different aspects:

- convergence of the algorithm: we measure the distance between the user models in the population and we show that this distance decreases; this is indicative of how the system learns the real user interests;
- learning the user models: we analyze the correlation between the evolution of the user models and the activities performed by the users of the system.
- quality of the proposed documents: the main aim of the proposed approach is that the more the users interact with the system, the more the quality of the documents proposed by the system increases. We have analyzed this aspect by investigating if the documents read by the users have been proposed by the UPR module or not.

IFeed was launched on September 1st, 2009. Since then, it has been running 24 hours a day and, as of December 10th, more than 500 users had signed up to this project. Not all the registered users have performed the same amount of activity. Therefore, for the purpose of evaluation, we have considered only the 50 most active users. For each user, we have considered his or her first 100 sessions. In each session, a user generally reads more than one document. A brief statistic analysis related to the users activities illustrates that the total number of documents read by users is 12,578, the average documents per user is 251.56 with a standard deviation of 120.79.

The feeds are grouped in 9 different categories based on different topics (*Politics, Economics, Science, Sports*, etc.). The considered users have been assigned to each category by observing their activities. For instance, if most documents read

² <http://www.genalgo.com/IFEEDINT>

by a user come from the science feeds, the user is assigned to the *Science* category. For each category, we have computed the graphs related to the users' activities in order to study the correlation between the behavior of the evolutionary algorithm and the way the users interact with the system. Moreover, we have analyzed the ability of the system to propose interesting documents to the users. Due to lack of space, in this paper we present and discuss the results obtained for one of the categories, namely *Politics*. However, the considerations below hold for the other categories as well. We will consider five users who have been assigned to the considered category. Nonetheless, the graphs presented represent the convergence of the algorithm with respect to all of the 50 most active users.

4.1 Creation of the Proposed Document Set

In this section, we describe in more detail how the UPR module works. This module is executed each time a user logs into the system. It aims at retrieving information about the user's profile and at creating a set of 20 documents to propose to the user. Here, for illustrative purpose, we consider a simple example. We assume to have a user with the models shown in Table 1.

Model 1		Model 2		Model 3	
Feed	Interest	Feed	Interest	Feed	Interest
1	0.79	1	0.60	1	0.32
2	0.24	2	0.90	2	0.09
3	0.55	3	0.33	3	0.21
4	0.12	4	0.01	4	0.49
5	0.89	5	0.77	5	0.94

Table 1. Examples of user models.

When the user logs into the system, the UPR module chooses randomly one among the models associated to the user, for example Model 2. Then, the interest of each feed in the model is normalized with respect to the sum of all interests of the chosen model. Subsequently, the feeds are ranked in descending order with respect to the feed interests. In order to present the user with documents distributed in a way to be proportional to the user's interests, we then calculate the number of such documents by multiplying the normalized interests by 20. Finally, the UPR module extracts and publishes documents until the limit of 20 documents has been reached. All the above considerations are illustrated in Table 2. We

Model 2 Norm.		Documents published	
Feed	Int. Norm	Coarse value	Published
2	0.34	6.8	7
5	0.30	6.0	6
1	0.23	4.6	5
3	0.13	2.6	3 $\rightarrow_{pub}$ 2
4	0.00	0.0	0

Table 2. Normalization and document publishing.

can notice that only two among the tree documents coming from Feed 3 will be published by the UPR module, because the limit of 20 documents has already been reached.

4.2 Learning Curve

In this section, the results related to the convergence of the system are presented. The system was set with a population of 30 individuals, the selection of individuals uses a tournament strategy with size 3, and the probabilities of crossover and mutation were respectively 0.75 and 0.10.

The value of 3 for the tournament size is to keep the selective pressure very low and to avoid premature convergence; with a probability of 0.75 for the crossover operator, we favor the exchange of user interests between the models and, finally, with a mutation probability of 0.10 we allow the system to explore more possible solutions. As said above, we have considered only the first 100 sessions for each user, each session corresponds to a generation of the evolutionary algorithm.

The formula used to calculate the sum of all the distances between all models is:

$$\text{Dist} = \sum_{k=1}^{m-1} \sum_{j=k+1}^m \sum_{i=1}^n |I_{F_i}^k - I_{F_i}^j|, \quad (13)$$

where n is the number of feeds in the system, m is the population size, $I_{F_i}^k$ is the interest in feed F_i of the model k , and $I_{F_i}^j$ is the interest in feed F_i of the model j . Thus, Dist represents the "overall distance" between the evolved user models present in the population and should be interpreted as an index of population diversity.

Figure 1 illustrates the learning curve obtained by analyzing the system's logs for the 50 users considered. We can notice that, for the first 30 generations, the algorithm converges quickly, then, after the 50th generation, the distance between models tends to remain constant. The value of zero is never reached due to the mutation operator. Indeed, the mutation operator introduces perturbations that maintain a small diversity even between individuals with similar interest degrees. Instead, a possible reason for the fact that the graph does not change after 50 generations is that we have supposed that the considered users are only interested in one category.

The models that have high interest degrees receive positive feedback. Therefore, such models have a higher probability of being chosen during the selection phase. After a given number generations the population is composed by similar individuals, therefore, the distance between the models is hardly conditioned by the perturbations applied by the mutation operator.

In general, since the interest for each feed is encoded as a separate gene in the genotype of a model, learning user interests gets harder as the number of feeds available to the users increase. If a user is interested in a restricted subset of feeds, however, it is relatively easy for the evolutionary algorithm to set the genes of the uninteresting feeds to zero and to focus on the few interesting feeds. This is why, when the number of feeds associated to each category is small, as it is the case with the *Politics* category here, the number of generations required by the system to learn satisfactory interest degrees is small too.

On the contrary, when users are interested in different categories, the convergence speed is slower than the situation

described above, because the evolutionary algorithm needs to search a larger space of candidate user models.

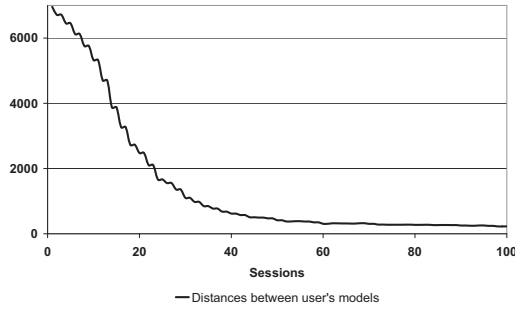


Figure 1. System's learning curve.

4.3 Statistics of User Activity

In this section we present some statistical data about the user models. This data represents the correlation between the way the users interact with the system and how the user models evolve as a result of these interactions. The presented values correspond to the average of the results obtained for the five users associated to the chosen category. In addition, for completeness, we illustrate in each graph the standard deviation of such results.

Figures 2, 3, and 4 show, respectively, the average, the maximum, and the minimum of the following values:

- the user's interest degrees at the initialization of the user models;
- the user's interest degrees after the last evolutionary generation considered;

Moreover, in Figure 5 we present the statistics of the documents read by the users for each feed. The x -axis represents the feed number, while the y -axis represents the percentage of read documents coming from the corresponding feed. The lines with markers represent the percent standard deviation calculated over the considered users.

We can notice that most of the read documents come from Feeds 1 and 14, which are the feeds associated with the *Politics* category considered in this example. However, the users also read, in a lesser proportion, documents coming from different categories, although we do not study this aspect in depth. We can observe that the average, the minimum, and the maximum value of the interest degrees related to those feeds are very similar. This is due to the ability of the system to learn the user's profiles. Indeed, when a user reads documents coming always from the same two or more sources alike, the models associated with the user's profile tend to assume similar interest degrees for those sources.

Moreover, we can observe that the more a user reads documents coming from a given feed, the less is the standard deviation associated with that feed. Indeed, for Feeds 1 and 14, the standard deviation is very small in every graph. The interest degrees of the other feeds exhibit a higher standard deviation than the two feeds cited above do. This happens for

two reasons. The first is that a small number of documents coming from those feeds has been read, so that the system does not have enough information to learn the exact interest degrees. The second is that the genetic operators constantly apply perturbations on all genes, thereby causing the seldom used interest degrees, which are less sensitive to the pressure of selection, to drift away from their optimal values in time. Surprisingly, Feed 18 too presents a small standard deviation. However, after a more in-depth analysis of the results we concluded that this case must be just a coincidence.

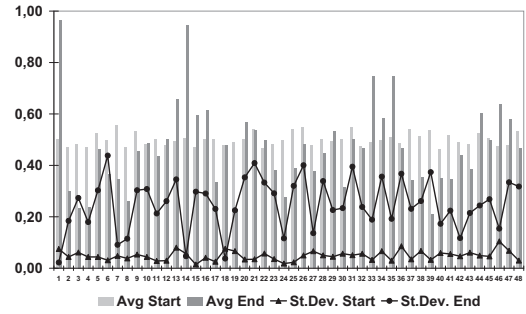


Figure 2. Average interest degrees for each feed calculated on the user profiles.

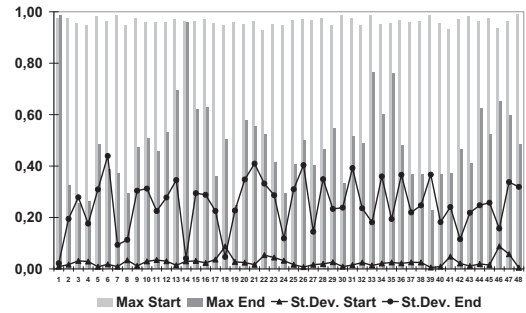


Figure 3. Maximum interest degrees for each feed calculated on the user profiles.

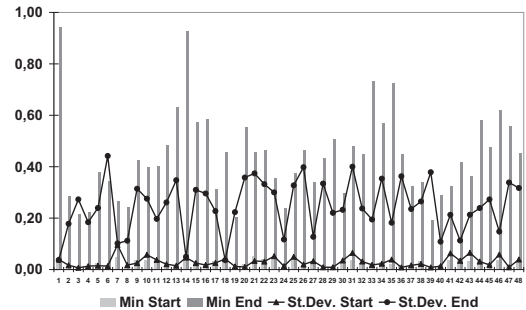


Figure 4. Minimum interest degrees for each feed calculated on the user profiles.

4.4 System Evaluation

In this section we discuss how the system improves the quality of the document set proposed to the users. As explained previously, every time a user logs into the system, the UPR module presents the user with a set of 20 documents. The aim of our approach is to present the users with documents that are as much as possible in line with their interests. To evaluate this aspect we have considered how many documents, out of those proposed to the user by the system, have been read by users with respect to the total number of documents read. To compare the five users considered above, we have taken the first 200 documents read by each user into account.

The results are shown in Figure 6. The graph represents the ratio between the number of documents read belonging to the set of the documents proposed by the system and the total number of documents read. Each point is calculated as follows:

$$R_j = \text{DRP}_j / j, \quad (14)$$

where DRP_j is the number of documents proposed by the system read when the j th document has been read and R_j is the value of the ratio after the user has read the j th document. We can observe that the more the users interact with the system, the more this ratio increases. This means that the system is able to propose an increasing number of documents that are interesting to the users. We can also notice that the system's behavior is not the same for all users. The reason behind that is quite obvious: the implicit feedbacks received by the system for each document read are not the same for all users; therefore, for each user, the time required by the system to learn the user's profile may vary.

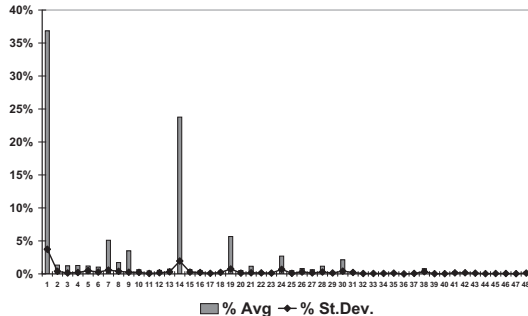


Figure 5. Statistics about the percentage of documents read for each feed.

5 Conclusions

This paper describes an approach to user model-based information retrieval which exploits a concept-based representation and a machine learning technique, namely evolutionary algorithms, to evolve models of user interests. The evolved models are used to propose documents to users, that are in line with their interests. The approach is also capable of dynamically tracking changing user interests.

Experiments have been performed on a real-world prototype newsfeed aggregator with search facilities called IFeed, to validate the approach with respect to two criteria: the first

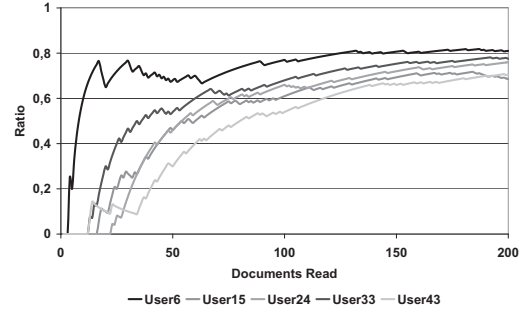


Figure 6. Trend of the quality of the set of the documents proposed by the system.

is the capability of learning user interests: the results show that the distance between the evolved models decreases as the user interacts with the system. The second is the increase of precision of the set of documents proposed to the users, indeed, the results show that, as the evolved models get closer to the real user interests, the users increase their activities on the set of documents proposed by the system.

References

- [1] M. Baziz, M. Boughanem, N. Aussenac-Gilles, and C. Christment. Semantic cores for representing documents in ir. In *SAC '05*, pages 1011–1017. ACM, 2005.
- [2] B. Crabtree and S. Soltysiak. Identifying and tracking changing interests. *International Journal on Digital Libraries*, 2(1):38–53, 1998.
- [3] K. A. DeJong. *Evolutionary Computation: A unified approach*. MIT Press, Cambridge, MA, 2002.
- [4] M. Dragoni and A. Tettamanzi. Evolutionary algorithms for reasoning in fuzzy description logics with fuzzy quantifiers. In *Proceedings of GECCO 2007*, volume 2, pages 1967–1974, London, UK, 2007.
- [5] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.
- [6] H. P. Giger. Concept based retrieval in classical ir systems. In *SIGIR '88*, pages 275–289. ACM, 1988.
- [7] C. Goldman, A. Langer, and J. Rosenschein. Musag: an agent that learns what you mean, 1996.
- [8] H. M. Haav and T. L. Lubi. A survey of concept-based information retrieval tools on the web. In *ADBIS'01, Advances in Databases and Information Systems*, pages 29–41. Vilnius "Technika", 2001.
- [9] D. Kelly and N. J. Belkin. A user modeling system for personalized interaction and tailored retrieval in interactive IR. In *Proceedings of the American Society for Information Science and Technology*, pages 316–325, 2002.
- [10] L. Shen, Y. Lim, and H. Loh. Domain-specific concept-based information retrieval system. In *Proceedings of the IEEE International Engineering Management Conference*, volume 2, pages 525–529, 2004.
- [11] S. Soltysiak and B. Crabtree. Automatic learning of user profiles — towards the personalisation of agent service, 1998.
- [12] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
- [13] L. A. Zadeh, G. J. Klir, and B. Yuan. *Fuzzy Sets, Fuzzy Logic and Fuzzy Systems: selected papers by A. L. Zadeh*. World Scientific, Singapore, 1996.
- [14] J. Zakos and B. Verma. Concept-based term weighting for web information retrieval. In *ICCIMA'05*, pages 173–178. IEEE Computer Society, 2005.