An Evolutionary Approach to Ontology-Based User Model Acquisition

Célia da Costa Pereira[1] and Andrea G.B. Tettamanzi[2]

[1] Genetica S.r.l.,
Via S. Dionigi 15, I-20139 Milano, Italy
dacosta@genetica-soft.com
[2] Università degli Studi di Milano,
Dipartimento di Tecnologie dell'Informazione,
Via Bramante 65, I-26013 (CR), Italy
andrea.tettamanzi@unimi.it

Abstract. In this paper we propose a new approach to User Model Acquisition (UMA) which has two important features. It doesn't assume that users always have a well-defined idea of what they are looking for, and it is ontology-based, i.e., we deal with *concepts* instead of *keywords* to formulate queries.

1 Introduction

Research in User Model Acquisition (UMA) has received considerable attention in the last years. The problem of information overload leads to a demand for automated methods to locate and retrieve information with respect to users' individual interests. Unfortunately, methods developed within information retrieval [7, 8] leave two main problems still open.

The first problem is that most approaches assume users to have a well-defined idea of what they are looking for, which is not always the case. We solve this problem by letting fuzzy user models evolve on the basis of a rating induced by user behavior. The second problem concerns the use of *keywords*, not *concepts*, to formulate queries. Considering words and not the concepts behind them often leads to a loss in terms of the quantity and quality of information retrieved. We solve this problem by adopting an ontology-based approach.

The approach described in this paper has been implemented and successfully tested in the framework of the Information and Knowledge Fusion (IKF) Eureka Project E!2235.

2 Data Representation

We deal with the problem of learning user interests from user behavior in a document retrieval system. Therefore, a suitable representation has to be found for documents, concepts and their relations, and user interests.

V. Di Gesú, F. Masulli, and A. Petrosino (Eds.): WILF 2003, LNAI 2955, pp. 25–32, 2006.

We assume concepts and their relations to be organized in a formal ontology [6, 9]. A lexical database like WordNet[1] or one of its derivates, will provide the link between words appearing in a document and the concepts they may refer to (i.e., their senses/meanings).

2.1 Document Representation

We have been inspired by the method proposed in [3], and we extended it to work with ontological concepts. In this framework, a document can be represented as a vector, whose components express the "importance" of every concept.

However, passing from an unstructured space of words seen as independent entities to an ontology of concepts, structured by the hierarchical *is-a* relation into a lattice, is not trivial. Suppose, for example, that we find in a document the three related concepts of *cat*, *dog*, and *animal*: now, *animal* is a super-class of both *cat* and *dog*. Therefore, mentioning the *animal* concept can implicitly refer both to *cat* and *dog*, although not so specifically as mentioning *cat* or *dog* directly would. Therefore, we need to devise a system to take this kind of interdependence among concepts into account when calculating levels of importance.

The main idea is to consider but the leaf concepts in the ontology, i.e., the most specific concepts only, as elements of the importance vector for a document. This choice can be justified by thinking that more general concepts (i.e., internal concepts) are implicitly taken account of through the leaf concepts they subsume, by "distributing" their importance to all of their sub-classes down to the leaf concepts in equal proportion [2] (this ensures that the components of a document vector are independent, or orthogonal, with respect to each other):

$$N(c) = \mathrm{occ}(c) + \sum_{\mathbf{c} \in \mathrm{Path}(c,\ldots,\top)} \sum_{i=2}^{\mathrm{length}(\mathbf{c})} \frac{\mathrm{occ}(c_i)}{\prod_{j=2}^{i} \|\mathrm{children}(c_j)\|},$$

where $N(c)$ is the count of explicit and implicit occurrences of concept c, and $\mathrm{occ}(c)$ is the number of occurrences of lexicalizations of c.

Since an ontology can contain a huge number of concepts (e.g., in the order of the hundreds of thousands), it is convenient to limit ourselves, for the purpose of calculating document importance vectors, only to the top-level ontology, whose size may be in the thousands of concepts [3]. Anyway, it is important to remark that the approach described below is absolutely independent of which subset of the ontology one decides to restrict to, and that what we call "leaf concepts"

[1] URL: `http://www.cogsci.princeton.edu/~wn/`.

[2] The assumption that the importance of an internal node is distributed to all subclasses in equal proportion is convenient in case of lack of further information on which subclass is more "typical" or "usual". However, taking that kind of information into account would significantly complicate the model and it is not clear what the gain in accuracy would be.

[3] Of corse if the ontology doesn't contain a huge number of concepts this limitation is not necessary.

can be defined in any way that is consistent with our premise, that is, that they form a set of mutually independent concepts (i.e., none of them subsumes, or is subsumed by, any other). The ontology lattice, cut this way, can be treated as a direct acyclic graph (*dag*), having $\top$ as its root.

The quantity of information info(c) given by the presence of some concept c in a document is given by

$$\text{info}(c) = \frac{N_{doc}(c)}{N_{rep}(c)},$$

where $N_{doc}(c)$ is the number of times a lexicalization of c appears in the document, and $N_{rep}(c)$ is the total number of its occurrences in the whole document repository.

Each document is then represented by a document vector I, whose jth component is

$$I(c_j) = \frac{info(c_j)}{\sum_{i \in \mathcal{L}} \text{info}(i)},$$

where c_j is a leaf concept, and $\mathcal{L}$ is the set of all leaf concepts.

We define similarity between documents on the basis of their vector representation, and we group them into clusters according with their similarity, in order to express user interests with respect to clusters instead of all individual documents. Any fuzzy clustering algorithm, like fuzzy c means [2], can serve this purpose. Furthermore, we characterize each cluster by a set of *key concepts*, i.e., the most important concepts in the vector representation of its member documents.

2.2 Representing User Interests

What we expect from a model of user interest is the ability to tell to which degree a given document or concept interests a user. Therefore, a natural abstract representation for a user interests is a fuzzy set of document clusters. We should observe that by knowing to which degree each given cluster belongs to the user interests, it is possible to derive the degree to which every document interests a user. We propose tree propositions for defining the interest of a document $I(d)$:

- the *optimistic* one which has been inspired by the optimistic utility proposed by Yager in [10].

  $$I(d) = \max_C \min(\mu_C(d), I(C)).$$

  Where $I(C)$ represents the interest of the user for the cluster C and $\mu_C(d)$ is the degree of membership of the document d in the cluster C. This definition says that only a small membership degree of a document to all very interesting clusters can decrease its interest degree. If the document doesn't belong to any cluster in the user profile then its interest is equal to 0.
- the *pessimistic* one which has been inspired by the pessimistic utility function defined by Dubois and Prade in [5] which at its turn has been inspired by the Yager's optimistic utility.

  $$I(d) = \min_C \max(1 - \mu_C(d), I(C)).$$

– the *mean value based* one

$$I(d) = \frac{\sum_C \mu_C(d) \cdot I(C)}{\sum_C \mu_C(d)}$$

In practice, a model of user interests is represented by a vector of membership values, one for each document cluster, i.e., of cluster interests.

3 Evolving User Models

The principle of our approach to UMA is the following. When a user logs in the system, there are two possibilities: it is either a new user or a known user. In the former case, the user will be asked to enter their personal data and a description of their interests (optional). In both cases, an evolutionary algorithm [1] is automatically set to run in the background, its objective being to optimize a model for the current user, with minimal user input.

Information gathered directly from the users at the time of their registration is used to generate an initial population of user models. From then on, it is the evolutionary algorithm which is responsible for tracking their interests as they interact with the system. Every action taken by a user in response to any proposition made by the system on the basis of a user model is interpreted as a rating of that user model.

3.1 Initialization

The first step concerns the case when a user is not yet known by the system. Our approach consists then in using the set of interest clusters which groups similar documents to initialize the list of interest clusters in the user profile. Summarizing, a user whose profile does not contain any group of interests (a new user), is asked (optionally) to describe, with key concepts, document titles, short phrases or others, their center of interest. The profile of the user is then set up by adding in the list of its interest groups, instances of clusters with characteristics similar "enough" to those requested by the user. To each of these groups of interests there is associated a degree corresponding to the importance the user attributes to that cluster. This importance depends on the interest degree of the documents in the cluster and their respective membership degree.

Let $\mathcal{U}$ be the set of the key concepts concerning the request introduced by the user, and $\mathcal{C}$ the set of key concepts associated to a document d. The importance $I(d)$ that this document has for the user is estimated by:

$$I(d) = \frac{|\mathcal{U} \cap \mathcal{C}|}{|\mathcal{U} \cap \mathcal{C}| + |\mathcal{U} \setminus \mathcal{C}| + |\mathcal{C} \setminus \mathcal{U}|} \tag{1}$$

As we said previously, the interest for a cluster C depends on the interest of documents in the cluster and their membership degree. Here, we define tree possibilities for computing this interest:

– the *optimistic possibility*

$$I(C) = \max_d \min(\mu_C(d), I(d)).$$

If all documents of C belong completely to C ($\mu_C(d) = 1 \; \forall \; d \in C$) then $I(C)$ corresponds to the degree of the most interesting document in C.

– the *pessimistic possibility*

$$I(C) = \min_d \max(1 - \mu_C(d), I(d)).$$

With this definition we can see that the less a document belongs to a cluster the less its interest degree influences the general interest of the cluster. If all documents of C belong completely to C ($\mu_C(d) = 1 \; \forall \; d \in C$) then $I(C)$ corresponds to the degree of the least interesting document in C.

– the *mean based possibility* which is simply computed using the mean weighted value of the document's interests.

$$I(C) = \frac{\sum_d \mu_C(d) \cdot I(d)}{\sum_d \mu_C(d)}.$$

Let $\mathcal{C}_i$ be the set of key concepts associated with documents d_i in the cluster C. The list of concepts associated to this cluster in the user profile contains only the concepts in $|\mathcal{U} \cap_i \mathcal{C}_i|$. We can see that if the user doesn't insert its preferences, the interests of all clusters are initially set to zero. Their set up are made with the evolution of the acquisition process.

The second step deals with the case in which the user is not new. In this case, the user interests are obtained automatically using an evolutionary algorithm to optimize a set of models (propositions) to the user. Summarizing, a user profile contains:

– the personal data of the user,
– a list of fuzzy clusters which interest the user and their respective documents,
– the degree to which each cluster interests the user (its importance for the user).

3.2 Representation

A user model is represented as a vector of membership degrees in $[0, 1]$, which describe the model's guess at the extent to which the user is interested in each document cluster. This is the most natural representation, given the abstract model of a user interests as a fuzzy set of document clusters.

3.3 Fitness Calculation

The evolutionary algorithm is on-line: each time the user interacts with the system, one of the models in the current population is randomly chosen to make its proposition to the user. If the user is satisfied (resp. unsatisfied), the fitness

of the chosen model increases (resp. decreases), as well as the fitness of all other models with the same proposition. The fitness of the other models is not changed. The on-line property of our algorithm poses some interesting problems, in that a precise calculation of fitness is not available for every individual at any moment. We invite the reader to see [4] for more details.

3.4 Genetic Operators

The algorithm is elitist, i.e., it always preserves a copy of the best individual unchanged. The *mutation* operator affects each individual with a probability inversely proportional to its fitness.

The *recombination* operator is uniform crossover: each child has a 0.5 probability of inheriting each gene from either parent. We tried several probability crossover rates and 0.5 is the probability which assures a non homogeneous population after few generations.

Deterministic tournament *selection* with a tournament size of 3 is responsible for selecting the pool of parents for the next generation. This tournament size value has been choose after many experiments. It avoids to have, at any generation step, an homogeneous population essentially constituted by the current best individual.

4 Validation of the Approach

The approach described above has been implemented and experimentally validated as follows. A simulation was set up whereby user interests are randomly created and used to simulate user interaction with the system. Precisely, initially a reference model is created. While evolving, the algorithm makes propositions, which are rated according to the reference model. This allowed us to effectively verify that the evolutionary algorithm was capable of learning user interests from user feedback. Figure 1 shows a graph of the distance between the reference model and those used by the system to make its propositions during one of the tests.

In a subsequent phase, a drift effect was applied to the simulated user interests, and the ability of the algorithm to effectively track changing user interests was experimentally proved.

5 Discussion

The evolutionary algorithm described above will not be always running. It will run only when a request is made by the user. In this case, once a feedback from the user is received, evolution will advance by one generation and then pause, waiting for the next user interaction.

This work is a significant step ahead toward the definition of a UMA approach suited for an ontology-based knowledge management framework, like the one

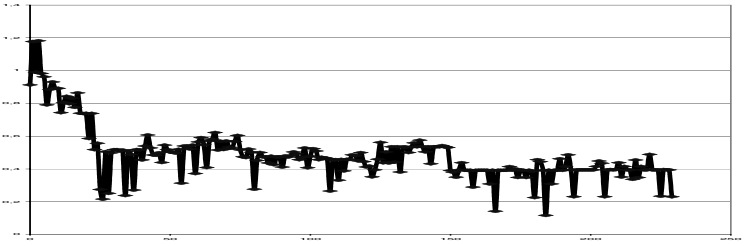

Fig. 1. Distance between the model to learn and the models obtained during the evolutionary process

developed in the IKF Project. This approach is novel, although inspired by several works in the literature.

We have first established the representation for the data of the problem. We have redefined the notion of document vector to consider concept, rather than keyword, occurrences in a document. Thus, a document is represented by a vector containing the importance of all the leaf concept obtained when considering the ontology only to a given depth. We have also defined a similarity function for two documents to use a fuzzy clustering algorithm.

A user model is essentially made up of personal data, a set of document clusters grouped according to their similarity and, for each of these document clusters, an associated degree corresponding to the interest of the user in that document cluster. The user models evolve in time tracking changes of user preferences. An evolutionary algorithm has been devised to obtain this behavior. The goal of that algorithm is to find, at each stage in evolution, the best models, i.e., the ones capable of making the best propositions to the user.

References

1. T. Bäck, D. B. Fogel, and T. Michalewicz, editors. *Evolutionary Computation (in two volumes)*. Institute of Physics, Bristol, Philadelphia, 2000.
2. J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.
3. B. Crabtree and S. Soltysiak. Identifying and tracking changing interests. *International Journal on Digital Libraries*, 2(1):38–53, 1998.
4. E. Damiani, A. Tettamanzi, and V. Liberali. On-line evolution of fpga-based circuits: A case study on hash functions. In A. Stoica, D. Keymeulen, and J. Lohn, editors, *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, pages 26–33, Pasadena, CA, July 19–21 1999. IEEE Computer Society.
5. Didier Dubois and Henri Prade. Possibility theory as a basis for qualitative decision theory. In *IJCAI*, pages 19–25, Montreal, 1995.
6. Nicola Guarino. Formal ontology and information systems. In Nicola Guarino, editor, *Formal Ontology and Information Systems: Proceedings of the first international conference (FOIS'98)*, Amsterdam, 1998. IOS Press.

7. Bernardo Magnini and Carlo Strapparava. Improving user modelling with content-based techniques. In *Proc. 8th International Conference on User Modelling*, pages 74–83. Springer-Verlag, 2001.
8. Andrea Rodriguez and Egenhofer. Determining semantic similarity among entity classes from different ontologies.
9. John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, Pacific Grove, CA, 2000.
10. R. Yager. An approach to ordinal decision making. *International Journal of Approximate Reasoning*, 12:237–261, 1995.