Courtz: An Agent that Pleases You (An Extended Abstract)

Claudia V. Goldman and Yishay Mor and Jeffrey S. Rosenschein
Institute of Computer Science
Hebrew University
Givat Ram, Jerusalem, Israel
ph: 011-972-2-658-5353
fax: 011-972-2-658-5439
email: clag@cs.huji.ac.il, yish@cs.huji.ac.il, jeff@cs.huji.ac.il

1 Introduction

Sometimes we have the feeling that "if we knew what we were looking for, we would know where to find it." Our objective in this project is to assist users in finding something, when they are not quite sure what that something is; or, even if they do know what they are looking for, they do not know how to explicitly describe it. As a case study, we have chosen the domain of university courses. A student is assumed to have some notion of his preferences over different courses, and additionally has constraints. However, this information is not necessarily sufficient for posing a query to a database system; in many ways it is *fuzzy* [1, 6] or incomplete.

Interface agents were defined in [7] as

> [C]omputer programs that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular application ... The metaphor is that of a personal assistant who is collaborating with the user in the same work environment.

Examples of interface agents include Letizia [5] and WebWatcher [2] which assist users while they browse the Web; and a News filter agent [3] which keeps its user updated with information that might be of interest to him.

Courtz is also a user interface agent, which uses tools from decision theory, fuzzy logic and probabilistic reasoning to try and learn the user's preferences, even when the user is not sure about them himself.

2 The Agent Architecture

Courtz is an agent that is accessible through the Internet. Its knowledge base is based on information available from the Hebrew University web site (http://www.cs.huji.ac.il/).

The user is presented with a form (i.e., an HTML form) that allows him to state his preferences and requirements. The user need not fill in every field, and he might even give inconsistent information.

This information is treated by Courtz as "soft" constraints: it assumes that the user will compromise constraints for whatever course is closest to his desires. For example, the user might ask for a three credit course, although he might also accept a two or four credit course. The underlying assumptions are that the user attaches different priorities to different aspects (e.g., subject, day, credit points) of the courses, and that his constraints over each aspect might be fuzzy.

After parsing the user's request, Courtz sorts the available courses by their utility with respect to that request. It then offers the user the K best courses, according to its current model of the user's desires. The user may now do one of the following:

- Accept one of the suggested courses.
- Update or refine his stated preferences.
- Re-rank the suggested courses.

If the user has not accepted an offer, Courtz uses the feedback received from the user to update its model of the user's preferences, and iterates.

2.1 Representation of the user's preferences

The objective of the representation of the user's request is to supply a *utility function* over the search space (i.e., the space of all possible courses). This utility function is approximated as a linear combination of field-wise utility functions, with respect to the request's fields. Each value submitted by the user in a particular field is translated into a *fuzzy membership function* over the domain of that field.

For example, our current implementation includes the following fields:

- Course times (days and hours)
- Teacher
- Subject
- Credit points

For any field, the user has a default null option (e.g., "anytime" for the time field). We want Courtz to "understand" that if a user specifies "Multi Agent Systems" as his preferred topic, he would also gladly accept a course on "Distributed AI," so such a specification would be translated into a function that assigns a membership value of 1 to "Multi Agent Systems", 0.99 to "MAS", 0.9 to "DAI", etc. Needless to say, this information cannot all be learned from the user: some of it must come from semantic knowledge of the domain supplied to Courtz (by the designer or by some other agent, e.g., Musag [4]).

As another illustration of the use of fuzzy functions, the user is presented with a choice of times such as "morning," "noon," "afternoon" and "evening." This makes specifying the request less tedious for the user, but more vague to Courtz. The nature of these concepts calls for a fuzzy representation: "noon" is a function with a peak around 12:00, decaying towards 15:00, and towards 10:00.

For each field, a distance function has to be defined, that computes the correlation between the (fuzzy) value of the request, and the corresponding attribute of a given course. Using these distance functions, we define a course's utility by:

$$\mathcal{U}(\mathcal{C}_|) = -\Sigma_i(\delta(R_i, C_{ji}) * \alpha_i)$$

where C_j is a given course, R is the user's request, and i runs over all the fields. α_i is the relative weight assigned to field i.

This model might differ from the user's true utility function in two ways: either the membership functions are wrong (e.g., by "4 points" the user means "not more than 4", while Courtz assumes "between 3 and 5"), or the weights assigned do not reflect the importance ascribed by the user to the different fields. This is where Courtz's learning mechanism is applied.

3 The Learning Mechanism

Courtz is involved in a two-phase learning process. On the one hand, Courtz needs to learn about the distance function between the values of the different fields and the values submitted by the user. On the other hand, Courtz also has to learn the relative importance of each field as seen by the user.

The distance functions are defined by the shape of the fuzzy function attributed to each value given by the user. Thus, learning these functions is actually a problem of learning fuzzy concepts. We will refer to this problem as the *FC-learning* problem.

The other part of the problem, learning the α values in the utility equation above, is a problem of learning a vector of weights in a linear combination. This is a relatively simple problem by itself, equivalent to the perceptron learning problem [8]. We will refer to this problem as the *weight learning problem.*

As a first step, we deal with each of the learning problems independently, i.e., attempt to solve the FC-learning problem as if the weights are correct, and solve the weight learning problem as if the concepts are accurate.

Two types of data are available to facilitate the learning process. As mentioned above, the user can refine his requirements, and can rerank the proposed courses. The first type of data, refinements of the request, can be used as a basis for explanation or induction-based learning. The reranking of the proposals supplies numeric information adequate for statistical learning techniques.

When the user refines some of the fields, he is doing so in reaction to Courtz's recommendation. Two things can be inferred from this: first, the refined fields are seen as important by the user. Refining them might indicate that the user feels they have not received enough attention. Secondly, the user signifies a discrepancy between his intention and Courtz's concept of it. For example, if the user chose "noon" as the course time, and now switches to "morning", he is indicating that Courtz's interpretation of "noon" is biased downwards.

The other type of information supplied by the user, the reranking of proposed courses, offers itself readily for statistical solutions to the weight learning problem. Courtz's utility function assigned K real numbers to the proposed courses. Assume course j was ranked best with $U(C_j) = x$, and course i was ranked fourth with $U(C_i) = y$. If the user reranks i as best, we know that $U^*(C_i) \geq x$. Similarly, if he reranks j as fifth, we may presume that $U^*(C_j) \leq y$. Thus, each reranked course supplies us with a data-point on which to run our learning algorithm.

References

[1] Zadeh Lofti A. Fuzzy sets. *Information and Control*, pages 338–353, 1965.

[2] Robert Armstrong, Dyne Freitag, Thorsten Joachims, and Tom Mitchell. Webwatcher: A learning apprentice for the World Wide Web. In *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*, 1995.

[3] Marko Balabanovic and Yoav Shoham. Learning information retrieval agents: Experiments with automated web browsing. In *Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments*, 1995.

[4] Claudia V. Goldman, Amir Langer, and Jeffrey S. Rosenschein. Musag: an agent that learns what you mean. In *The Second International WWW Conference on Practical Applications of Intelligent Agents and Multi-Agent*, 1996, submitted.

[5] Henry Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Quebec, 1995.

[6] Tsippy Lotan. *Modeling Route Choice Behavior in the Presence of Information Using Concepts from Fuzzy Set Theory and Approximate Reasoning.* PhD thesis, Dept. of Civil and Environmental Engineering, Massachusetts Institute of Technology, August 1995.

[7] Pattie Maes. Social interface agents:acquiring competence by learning from users and other agents. In Oren Etzioni, editor, *Software Agents - Papers from the 1994 Spring Symposium*, pages 71—78. AAAI Press, 1994.

[8] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, Cambridge, MA, 1969.