Trainable Sofware Agents

Yolanda Gil

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
gil@isi.edu

Abstract

Software agents usually take a passive approach to learning, acquiring information about a task from watching passively what the user does. Examples are the only information that the user and the learner exchange. The induction task left to the agent is only possible with pre-defined learning biases. Determining a priori the language needed to represent the necessary criteria for performing a task may not always be possible, especially if these criteria must be customizable. The focus of trainable software agents is on acquiring learning biases dynamically from users, being active in asking questions when it is not clear what to learn. The user's input to the learning process goes beyond concrete examples, providing a more cooperative framework for the learner.

Introduction

In order to be of widespread use, software agents should facilitate fast task automation and be flexible, evolvable, and adaptable. End users should be able to correct, update, and customize the system's knowledge through interfaces that support interactive dialogues. Current systems that acquire knowledge from a user take a passive approach. Programming by demonstration interfaces acquire programs from user's examples [Cypher, 1993]. Other systems learn from a teacher's instructions [Huffman and Laird, 1993]. Learning apprentices acquire knowledge by analyzing a user's *normal* interaction with a system [Dent *et al.*, 1992, Mitchell *et al.*, 1990, Wilkins, 1990]. Some systems learn from solutions given by the user, others display a proposed solution that the user can accept, correct, or rate [Baudin *et al.*, 1993, Dent *et al.*, 1992, Maes and Kozierok, 1993]. The advantage of this passive approach is that it is non-intrusive. Another advantage is that they are very easy to use by non-programmers, because examples are a natural way for communicating knowledge. However, it is hard to automatically generalize from examples and acquire knowledge that applies to new situations. It is also hard to write algorithms that can extract from the examples knowledge about what features of the example are important, what new abstract features that do not appear in the example need to be defined, what combinations of those features are relevant for defining criteria for the task, and what the user's preferences are. We claim that extracting this knowledge automatically is not the only viable option.

We are starting a research project to build *trainable agents*, systems that improve their performance on a task both autonomously and in cooperation with a user. Trainable agents are *proactive* learners, i.e., they can resort to asking questions of the user when autonomous learning reaches an impasse. They can learn autonomously by observing the user performing a task, and engage in an interaction with the user when it is not clear what to learn.

Proactive learners have been investigated in various areas. Results in computational learning theory show that they are faster and more effective learners [Angluin, 1987]. Other research investigates what sequences of queries to an external oracle improve the performance of a learning system, based on the information content of each possible query [Rivest and Sloan, 1988, Ling, 1991]. In learning from examples, there are some results on different strategies to request examples from a teacher, and how the order and nature of the examples affects the effectiveness of a learning algorithm [Subramanian and Feigenbaum, 1986, Ruff and Dietterich, 1989]. Learning from experiments allows learners to propose concrete questions to the environment and have control over the information they receive about the world [Mitchell *et al.*, 1983, Cheng, 1991, Hume and Sammut, 1991, Shen and Simon, 1989, Gil, 1993]. Queries, examples, and experiments correspond to questions to ask a user in a trainable agent. Although user in-

teraction brings up different issues, we believe their results will be very useful for our work.

Intended Capabilities of Trainable Agents

Trainable agents will be able to do the following:

- Perform a task to the best of its available knowledge;
- Acquire knowledge relevant for the task by direct instruction from the user;
- Autonomously learn new knowledge whenever possible;
- Initiate interaction with the user when additional information is needed for learning, but only resorting to interaction when it is cost effective in order to avoid placing excessive burden to the user;
- Interact with a user by asking relevant questions that are easy to answer, and suggesting answers that the learning algorithm can make most sense of.

A trainable agent can:

- Become increasingly more trusted by the user, evolving its interaction from apprenticeship, to cooperation, to final autonomy;
- Become increasingly more competent and more robust at performing the task, iteratively improving the quality of its performance by learning from users' corrections;
- Adapt to a user's personal needs and preferences;
- Be used for a task almost immediately, since it learns through use without extensive preprogramming, actively trying from the beginning to help users perform a task to the best of its available knowledge.

Trainable agents can perform structured routine tasks according to criteria that can be explicitly represented and are well understood by a human. In particular, trainable agents can be used for tasks involving classification, filtering, and marking of information. Figure 1 shows a schematic representation of such a trainable agent. We plan to use Bellcore's SIFT tool for email classification as our performance system.

A Scenario of the Interaction

Consider the task of classifying e-mail messages into folders. The agent is given some rough initial criteria useful for the task, and no knowledge about the preferences of the particular user. Given an email message described by some features (i.e., sender's name, sender's address) and a set of folders, the system learns additional features and a set of criteria (rules) to assign the object to a class. The goal of the system is to learn more about the task as well as to tailor its behavior to the user by learning through the interaction.

Given an input, the agent uses the current criteria to suggest a solution. If the user agrees, the system assimilates this episode as evidence for correctness of the criteria and runs the learning algorithm to see if information can be learned from the new input. If the user disagrees by showing a different solution, the trainable agent detects a fault in its current knowledge and an opportunity for learning.

Once learning is triggered, the system first tries to repair its knowledge autonomously by running a learning algorithm with the user's solution. If learning is possible, the result is a new set of criteria that would allow the system to produce automatically the user's solution in the future. However, in some cases, the system's limited knowledge may produce contradictions that prevent autonomous learning. For example, the solution indicated by the user recommends a course of action that violates one of the existing criteria. The system's criteria must be corrected, but autonomous learning failed. There is a need for the user's direct intervention for learning, and the goal of the system is to acquire the knowledge missing in its current criteria.

To initiate the user's intervention, the system presents the violated criteria as well as past scenarios in which those criteria produced a valid solution. To support the user's intervention, the system also proposes sensible options for criteria correction. If the user's response still creates conflicts, the system proposes what-if questions to the user. These will consist of hypothetical, partially specified inputs that concentrate on features that are currently raising the conflicts. The process iterates until the system obtains from the user the information needed to fix the knowledge deficiency initially detected. Since one important consideration is to minimize the burden on the user, the system needs to consider cost/benefit analysis before initiating an interaction.

The trainable agent will acquire new knowledge relevant for the task in the form of:

- new primitive features, i.e., those that appear in the input (e.g., the string "AAAI" in the subject line)
- new abstract features, i.e., those derived from primitive features (e.g., the concept "faculty-member")
- new target classes (e.g., creating a new folder for AAAI related messages)
- relations between concepts (e.g., faculty-member and student are disjoint)
- new definitions for existing concepts (e.g., part-time-student and student-worker)
- criteria for classification based on all of the above (e.g., when "AAAI" is part of the subject line, put message in the AAAI folder).

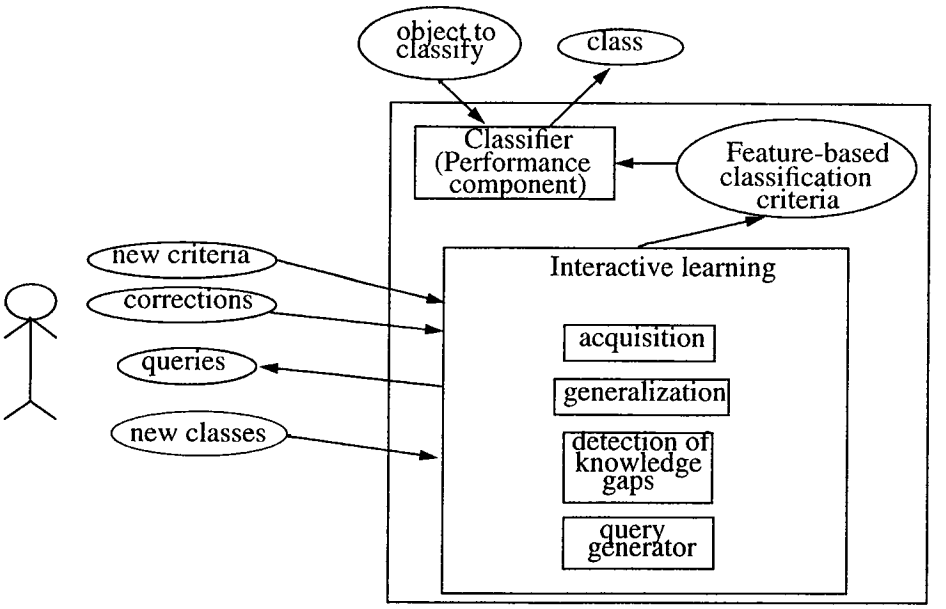

Figure 1: A schematic representation of a trainable agent.

The Challenge

Our goal is to build trainable agents that learn to do a task by acquiring knowledge both autonomously and interactively about features and criteria that specify how the user wants a task to be performed. Several technical problems need to be addressed:

- Knowledge assimilation: how can a user specify new features and classes so they can be incorporated into the existing knowledge.
- Initiating user interaction: under which conditions is it preferable for the system to seek interaction with the user to overcome learning impasses
- Minimizing interaction: what questions and in what sequence lead more effectively to acquiring the missing knowledge
- Support of interaction: how to ask questions, including options for possible answers to ease the interaction. We plan to develop a model to generate possible answers based on the expectations of the learning system for possible corrections

An underlying key issue is how to extract knowledge from concrete examples that can be applied in novel situations. Generalizations are only possible when the learner is given a *bias*, i.e., a language of patterns that abstracts from the concrete values that appear in the examples. Biases are not only needed but desirable [Mitchell, 1990]. In most systems, the generalization bias is hand-coded in a domain model [Baudin *et al.*, 1993, Dent *et al.*, 1992] and can only be changed by a system developer. Other systems avoid the need for a bias by operating at the example level using case-based techniques [Bareiss *et al.*, 1989, Maes and Kozierok, 1993, Cypher, 1993]. In such cases, the system needs to figure out which past example is most relevant to the new situation. This requires defining indexing schemas and similarity metrics that also require effort from the system's designer. Our trainable agents would be unique in their ability to acquire the learning bias dynamically from the user. This would reduce dramatically the effort needed to start the automation of an application and to evolve, maintain, and adapt to a task. In conclusion, we expect trainable agents to be key contributors to the ubiquity of software agents and other intelligent systems.

Acknowledgements

Discussions with Cecile Paris and Bill Swartout have helped a great deal to clarify the ideas presented in this paper. I would like to thank Kevin Knight, Tom Mitchell, Paul Rosenbloom, and Milind Tambe for their helpful comments and suggestions about this new line of work. Finally, I would also like to thank Sheila Coyazo for her prompt support on the keyboard.

References

[Angluin, 1987] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319--342, 1987.

[Bareiss *et al.*, 1989] Ray Bareiss, Bruce W. Porter, and Kenneth S. Murray. Supporting Start-to-Finish Development of Knowledge Bases. *Machine Learning*, 4(3/4):259--283, 1989.

[Baudin *et al.*, 1993] Catherine Baudin, Smadar Kedar, Jody Gevins Underwood, and Vinod Baya. Question-based aquisition of conceptual indices for multimedia design documentation. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, 1993.

[Cheng, 1991] Peter C-H. Cheng. Modelling experiments in scientific discovery. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[Cypher, 1993] Allen Cypher. Eager: Programming Repetitive Tasks by Demonstration. In Allen Cypher, editor, *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, 1993.

[Dent *et al.*, 1992] Lisa Dent, Jesus Boticario, John Mcdermott, Tom Mitchell, and David Zabowski. A personal learning apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992.

[Gil, 1993] Yolanda Gil. Efficient domain-independent experimentation. In *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, 1993.

[Huffman and Laird, 1993] Scott B. Huffman and John E. Laird. Learning procedures from interactive natural language instructions. In *Proceedings of the Tenth International Conference on Machine Leaning*, Amherst, MA, 1993. Morgan Kaufmann.

[Hume and Sammut, 1991] David Hume and Claude Sammut. Using inverse resolution to learn relations from experiments. In *Proceedings of the Eighth Machine Learning Workshop*, Evanston, IL, 1991.

[Ling, 1991] Xiaofeng Ling. Inductive learning from good examples. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[Maes and Kozierok, 1993] Pattie Maes and Robyn Kozierok. Learning interface agents. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, 1993.

[Mitchell *et al.*, 1983] Tom Mitchell, Paul Utgoff, and Ranan Banerji. Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine Learning, An Artificial Intelligence Approach, Volume I*. Tioga Press, Palo Alto, CA, 1983.

[Mitchell *et al.*, 1990] T. M. Mitchell, S Mahadevan, and L. Steinberg. LEAP: a learning apprentice for VLSI design. In *Machine Learning: An Artificial Intelligence Approach*, volume 3. Morgan Kaufmann, San Mateo, CA, 1990.

[Mitchell, 1990] Tom M. Mitchell. The Need for Biases in Learning Generalizations. In Jude W. Shavlick and Tom G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990.

[Rivest and Sloan, 1988] Ron L. Rivest and Robert Sloan. Learning complicated concepts reliably and usefully. In *Proceedings of the Workshop on Computational Learning Theory*, Pittsburgh, PA, 1988.

[Ruff and Dietterich, 1989] Ritchey A. Ruff and Thomas G. Dietterich. What good are experiments? In *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, NY, 1989.

[Shen and Simon, 1989] Wei-Min Shen and Herbert A. Simon. Rule Creation and Rule Learning through Environmental Exploration. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.

[Subramanian and Feigenbaum, 1986] Devika Subramanian and Joan Feigenbaum. Factorization in experiment generation. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, 1986.

[Tecuci, 1992] Gheorghe D. Tecuci. Automating Knowledge Acquisition as Extending, Updating, and Improving a Knowledge Base. *IEEE transactions on Systems, Man and Cybernetics*, 22(6):1444--1460, 1992.

[Wilkins, 1990] David C. Wilkins. Knowledge Base Refinement as Improving an Incorrect and Incomplete Domain Theory. In *Machine Learning: An Artificial Intelligence Approach, Vol III*, pages 493--513. Morgan Kaufmann, San Mateo, CA, 1990.