Building Intelligent Agents

An Apprenticeship Multistrategy Learning Theory Methodology Tool and Case Studies

by Gheorghe Tecuci

Academic Press, 1998, 315 p.+ XV
ISBN 0-12-685125-5

Building *intelligent* agents is one of the central goals (if not THE goal) of Artificial Intelligence. Unfortunately, the first attempts made in the 60's and the 70's failed due to the enormous difficulties put by such an enterprise. In the meantime, however, AI has witnessed important advances in its various subfields and has also become mature enough to address the problem of integrating these advances in practically useful systems.

The book presents a theory of and a methodology for building intelligent agents, together with case studies. The approach made consists in combining machine learning and knowledge acquisition for building intelligent agents. The knowledge acquisition aspect related is that of the knowledge engineer teaching the agent how to perform domain-specific tasks. Teaching is done by giving the agent examples and explanations, as well as by supervising and correcting its behavior. The approach presented in the book is called "Disciple". The "Disciple " approach limits the knowledge engineer's implication in the process of building an intelligent agent. The agent can help the user perform his tasks, may perform tasks on user's behalf, monitor events or procedures for the user, help users collaborate.

Current approaches to building intelligent agents are knowledge acquisition approaches and machine learning approaches.The book rightfully stresses the fact that these are complementary approaches that should be combined in a practically useful system, since, if alone, none of them can overcome the problems that occur in practice. In the knowledge acquisition approaches, the agent is built by a knowledge engineer. The knowledge engineer interacts with a domain expert to understand how the expert solves problems and what kind of knowledge he makes use of. He chooses the representation of knowledge , builds the inference engine, elicits knowledge from the expert, conceptualizes it and represents it in the knowledge base.

In the machine learning approaches, the agent employs learning methods to learn all of its knowledge from data or experience. Machine learning has focused on developing autonomous algorithms for acquiring knowledge from data and for knowledge compilation and organization.The goal is to improve a knowledge base agent's competence and efficiency in problem -solving. Machine learning research assumes that, before any learning takes place, there already exists a representation language and background knowledge.

The machine learning approach has produced basic learning strategies such as: empirical inductive learning from examples, explanation based learning, analogical learning, abductive learning, conceptual clustering, quantitative discovery, reinforcement learning, genetic algorithm-based learning, neural network learning.

Knowledge acquisition and machine learning share a common goal, i.e. to develop a knowledge base, but under complementary approaches. They cover all the three phases of knowledge base development process: systematic elicitation of expert knowledge, knowledge base refinement and knowledge reformulation .

There is a growing awareness within these two research communities that a better approach to building intelligent agents is through an integration of different knowledge acquisition and learning methods.

In **Chapter 1** the author reviews and contrasts the main knowledge acquisition, machine learning and integrated approaches to building intelligent agents.

Chapter 2 gives a general description of the Disciple approach for building intelligent agents

and illustrates it with two cases: an agent developed by an expert to be his assistant and an agent developed by an expert to be used by non-expert users. "Disciple" is a multistrategy learning approach for developing intelligent agents. An expert teaches the agent how to perform domain-specific tasks by giving the agent examples and explanations as well as by supervising and correcting its behaviour.The expert may teach the agent how to solve a certain type of problem by providing a concrete example, helping the agent understand the solution, supervising the agent as it attempts to solve analogous problems, and correcting its mistakes. The agent will be guided in learning complex problem -solving rules, and in extending and correcting its knowledge base. This process is based on a co-operation between the agent and the human expert; the agent helps the expert express his knowledge using the agent's representation language, and the expert guides the learning actions of the agent.The Disciple approach creates a natural environment for knowledge acquisition from the human expert.

The Disciple approach is implemented in the Disciple learning agent shell. The shell consists of a learning engine and an inference engine.

Two examples of agents built using Disciple are given: an assistant to the expert or an agent to be used by non-expert users. A first example is an assistant to the expert, more precisely a manufacturing assistant that collaborates with the expert in drawing up plans for loud-speaker manufacturing.The expert is both the developer and the user of the agent. A second example of an agent (developed by an expert for non-expert users) is that of an assessment agent that is taught by an educational and history expert to interact with history students (its non-expert users).The agent will generate tests to be solved by students, and could provide them with hints, answers and explanations.

The next two Chapters of the book present in detail the concept-based knowledge representation of the Disciple approach, as well as the problem - solving, knowledge acquisition and learning methods.

Chapter 3 describes in detail the knowledge representation and reasoning mechanisms of the Disciple agents. The knowledge representation used is well-suited for knowledge acquisition and learning.

"Disciple" agents use a hybrid knowledge representation, where domain objects are represented using semantic networks, and problem-solving strategies are represented using rules.These two different knowledge representation formalisms are combined in such a way as to compensate for each other's weaknesses and to take advantage of their strengths and complementariness.The objects are described in terms of their properties and relations and are organized according to *'more-general-than'* relation, thus forming a hierarchical semantic network.

The rules are expressed in terms of object names, properties and relations.The meaning of the rules depends on the application domain. These rules may be: inference rules for inferring new properties and relations; general problem-solving rules, such as the rules that indicate the decomposition of complex problems into simpler subproblems or action models that describe the actions that could be performed by an agent in terms of their preconditions, effects and involved objects.

Chapter 4 presents the knowledge acquisition and learning methods of "Disciple" agents that are used to build the knowledge base.These are the methods of systematic elicitation of knowledge for developing an initial knowledge base, the rule-learning method based on explanations and analogy, and the rule refinement method based on analogy, experimentation, and explanation-based and example-based induction. It also presents the exception-based refinement of the knowledge base that employs knowledge discovery and elicitation.

The main goal of knowledge elicitation in Disciple is the manual construction of an initial knowledge base to be extended and improved through apprenticeship and multistrategy learning and through further guided elicitation. During the initial knowledge elicitation phase, the domain expert has, assisted by a knowledge engineer, to define main knowledge structures of the application. The result of the initial knowledge elicitation phase is a semantic network of object concepts and instances.This semantic network is likely to be incomplete and partially incorrect.

"Disciple" agents learn PVS (plausible version space) rules from examples and explanations. The learned rule may represent a decomposition rule, an inference rule, or any type of *IF-THEN* rule.

As a result of the knowledge elicitation and rule learning processes, the knowledge base of the agent will consist of a semantic network and a set

of PVS rules.The semantic network was created by the expert who defined all the concepts that he could easily express.The rules learned by the agent are PVS rules. The lower bound of each PVS rule covers only one instance and the upper bound covers instances that are only expected but not guaranteed to be correct,because they are based on analogical reasoning.Therefore these rules are incomplete and possibly partially incorrect.

The knowledge base may also contain rules which are manually defined by the expert or are imported from an external knowledge base.These rules could the same be incomplete and partially incorrect.

The goal of rule refinement is to improve the rules as well as to extend and correct the semantic network until all the instances of the rules in the semantic network are correct. The rule refinement methods apply to PVS rules; the imported rules and the rules manually defined by the expert are first transformed by the agent into PVS rules.The basic idea is to interpret the rule's condition as a plausible lower bound condition by using the generalization method.

Chapter 5 presents the methodology for building intelligent agents by using the "Disciple" learning agent shell that implements the methods from the previous Chapters. It shows how a user interacts with the Disciple learning agent shell to teach and train it to become a domain-specific agent. This Chapter presents in more detail the architecture of the Disciple learning agent shell. The Disciple shell has a modular architecture which facilitates the development of learning agents for a variety of domains. It consists of four main components:

- a knowledge acquisition and learning component for developing and improving the knowledge base;
- a basic problem-solving component with facilities used by the knowledge acquisition and learning component;
- a knowledge base manager which controls access and updates to the knowledge base;
- an empty knowledge base to be developed for the specific application domain.

In addition, there are two other components that need to be developed and integrated with the Disciple shell to form an agent that performs specific tasks in an application domain:

- a graphical user interface which supports specialized knowledge elicitation and agent operation;
- a problem-solving component which provides the specific functionality of the agent.

The main purpose of the knowledge acquisition and learning component is to assist the expert during the development of the knowledge base.This component contains modules for knowledge elicitation, rule learning and rule refinement.

The problem-solving component of the shell provides basic capabilities which support the inferential processes necessary for rule learning and refinement and the basic operation of the agent.There are basic operations that can be used by a domain-specific problem-solver: transitivity of certain relations, inheritance of features in a semantic network,network matching, rule matching and example generation.

The knowledge base under development will be incomplete and possibly partially incorrect.The basic function of Disciple's knowledge base manager is to support this development process, providing a means to update the evolving knowledge base while ensuring consistency between knowledge elements.

This Chapter makes the transition to the remaining chapters of the book. Each of the last four chapters presents a case study of using the Disciple methodology and shell to develop intelligent agents for complex domains.

Chapter 6 presents in detail an application of the Disciple methodology to the building, training and using of two agents that generate history tests to assist in the assessment of student's understanding and use of higher-order thinking skills. The agents are representative of the class of agents built by an expert (in this case an educational and history expert) for user groups (in this case history teachers and students). One of the assessment agents is integrated with the MMTS (Multimedia and Thinking Skills) educational system, creating a system with expanded capabilities, called IMMTS (Intelligent Multimedia and Thinking Skills).The other agent is a stand-alone agent that is used independently of the MMTS software.

Chapter 7 presents a statistical analysis assessment and support agent, a Disciple agent that is integrated in an introductory, university level, science course and can be accessed on the Internet through a web browser. The course, called 'The Natural World', introduces students to the world of science using collaborative assignments and problem-centered group projects.The course deals with three issues:

diseases, evolutionary history and nutrition.The focus of the course is on analysis of documents and data,and integration of biological knowledge, statistical methods and historical and public health perspectives. The agent has three functions. First it can be used as a traditional test generator. Second the agent is integrated in the documents accessed by students, thereby making the learning process an active one. Finally, the agent can be used as an assistant to students as they work through their assigned projects.Students are required to include some data analysis to support their argument, and so students can use the statistical analysis assessment and support agent to help them extract all possible relevant information from a particular data set, and support their analysis of the data.

Chapter 8 presents a case study of using the Disciple approach to develop an agent that behaves as a personal assistant to the user.The main feature of this agent is that it is under constant supervision of the user who is both the developer of the agent and the beneficiary of its services.The agent is continuously supervised and customized by the user according to the changing practices in the user's domain, as well as the needs and the preferences of the user.The human designer and the assistant create designs together. The designs are initially created by the designer who is thereby teaching the agent. Gradually, the assistant learns to design, becoming a more useful assistant. This process constantly increases the capabilities of the design team consisting of the human designer and the computer assistant.

Chapter 9 presents the application of the Disciple approach to build automated agents for distributed interactive simulations of military exercices. The Sections of the Chapter illustrate the process of building and training an agent to behave as a virtual company commander in ModSAF (Modular Semi-Automated Forces) .This agent will receive orders to accomplish certain missions and will have to generate a plan of actions that will accomplish the mission.

We find the book useful to a large number of readers from research, industry and academia, as it fills in an important gap in the current literature on intelligent agents. Most books published in this area describe either very simple agents (that can hardly be called "intelligent" by current standards), or mathematically complex theories (using for instance combinations of branching time and modal logics), without any implementation solutions, thereby leaving the reader with an uncomfortable feeling. The present book strikes a nice balance between the theoretical soundness and elegance of the approach and its practical applicability. This is demonstrated not only by presenting the Disciple shell, but also by describing its use in practically interesting applications.

The main merit of the book from a conceptual point of view consists in using the complementariness of knowledge acquisition methods and machine learning techniques in order to take advantage of their strengths and to compensate for each other's weaknesses in the process of building intelligent agents. This aspect cannot be overestimated, as it significantly improves the efficiency and quality of the systems built with the Disciple shell.

From a technical point of view, the book is original also by integrating a form of expert-guided explanation-based learning, a kind of learning by analogy and experimentation (to automatically generate additional examples to be shown to the expert), and a kind of empirical induction (to learn from these examples). Disciple was one of the first attempts to integrate machine learning and knowledge acquisition.

The agent development methodology presented in the book can be of interest to practitioners as well. Also, since it integrates various key methods and techniques of Artificial Intelligence in general as well as Machine Learning and Knowledge Acquisition in particular, the book could be successfully used as a textbook in university curricula.

In conclusion, this book is a 'must-read' for anyone interested in building intelligent software systems and agents, as well as for researchers in AI, Machine Learning and Knowledge Acquisition.

We owe thanks to the Academic Press for enabling our contact with the book, and appreciation for their competence and taste in doing publishing work.

Monica Trifas