Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

"Paradigms of AI Programming" in Python

Daniel Connelly
Google Inc.
Dienerstraße 12, 80331 Munich, Germany
dconnelly@google.com

Ashok K. Goel
Design & Intelligence Laboratory, School of Interactive Computing, Georgia Institute of Technology
85 Fifth Street NW, Atlanta, GA 30308
ashok.goel@cc.gatech.edu

Abstract

Norvig's (1992) Paradigms of AI Programming is an important book for learning about AI programming. However, the book uses Common Lisp as the programming language, which is less popular now than in 1992. Thus, we have translated many classical AI programs described in the book into Python, a more commonly used language. We have also documented the programs and offered them as a resource in a course on knowledge-based AI.

Learning AI programming is a major part of learning AI

Since the inception of AI several decades ago, the AI community has developed a standard collection of techniques that can be applied to a wide range of situations. Because most of these models and techniques involve at least some computation, computer programming and simulation has been a very powerful and fruitful tool for AI researchers. As a result, AI has contributed to and benefited from other fields of computing such as data structures and algorithms, programming languages, and software engineering, and AI research and development teams typically include skilled computer programmers.

Mastering the standard AI models and techniques forms an important part of education in AI. An introductory AI course in a computer science program may require students to complete several programming assignments that apply standard techniques from problem solving, knowledge representation, learning, etc. These assignments provide students not only an opportunity to improve their computer programming and software engineering skills, but also provide a toolbox of solutions to common problems in developing intelligent systems.

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Peter Norvig's Paradigms of AI Programming: Case Studies in Common Lisp

Norvig's (1992) Paradigms of AI Programming: Case Studies in Common Lisp (PAIP for short) is an important book for learning about AI programming. The book surveys several historically significant AI programs and reviews many of the most important AI techniques by developing significant and complete computer programs that implement those techniques. In addition, the book introduces some of the most useful and important techniques in the practice of computer programming, including program design, automated testing, debugging, benchmarking, metaprogramming, object-oriented programming, and language implementation. This combination of hands-on AI learning with software engineering practice makes PAIP a useful book for intermediate-level students as well as working professionals.

Since the publication of PAIP in 1992, the popularity of Lisp (and Common Lisp) as a programming language has declined, and most students now have few opportunities to learn the language. At one time, Lisp played a central role in introductory courses in computer science in the form of Scheme, but now it has been mostly replaced by similar courses using Python. Further, Lisp's traditional role in AI courses too has been mostly replaced with Python or Java.

PAIP in Python

Thus the first author (Connelly) undertook a project with the second author (Goel) to work through PAIP and convert some of its most useful programs into Python. Our goal was both to expand the accessibility of PAIP and provide an educational resource to AI students. Connelly worked on this project over four months, and wrote Python implementations of the following programs:

- Basic search algorithms, including breadth-first and depth-first search, beam search, shortest-path computations, and pathfinding algorithms.
- General Problem Solver (GPS): a framework for applying means-ends analysis to any problem that can be decomposed into a system of goals and state transitions.
- Eliza: an interactive "psychiatrist" program that uses pattern matching and a rules database.
- Emycin: an expert system framework leveraging backwards chaining, probabilistic reasoning, and a rules database.
- Othello: a game-playing agent leveraging search algorithms and search trees.
- Prolog: a logic programming library and language interpreter for expressing and solving logic problems. Only the basic logical operations needed to express computation are implemented; none of the usual extensions found in commercial Prolog systems, such as built-in lists or syntactic sugar, are included.

Each program in PAIP in Python is accompanied by a full suite of automated test cases along with several complete examples that demonstrate possible uses. Further, each program is documented in a literate programming style, which incorporates prose expositions of the program development in the form of code comments alongside the executable program. In this way each program is a self-contained introduction to an AI topic that also happens to be executable. This follows the spirit advocated by Abelson, Sussman & Sussman (1984) in Structure and Interpretation of Computer Programs: “programs must be written for people to read, and only incidentally for machines to execute."

The entire project has been open-sourced and is hosted on the popular GitHub project hosting site. It can be found at https://github.com/dhconnelly/paip-python and contributions are welcome.

Course on Knowledge-Based AI

Connelly was inspired to work on the PAIP in Python project by a Georgia Tech course CS 7637: Knowledge-Based AI (KBAI) he took with Goel in Fall 2011. KBAI is a 3-credit semester-long graduate-level course that is offered each fall term. The Georgia Tech course catalog describes the contents of CS 4635/7637 as “Structured knowledge representation; knowledge-based methods of reasoning and learning; problem-solving, modeling and design.” The course is taught from the perspective of cognitive systems that takes the goals of AI as both building intelligent systems and understanding human intelligence. Thus, the course covers more traditional AI topics that leverage knowledge. These topics led to specific choices of PAIP programs to translate into Python. Some of these knowledge-based systems are very complex, and it is not feasible for most students to implement more than one or two during a semester-long course such as KBAI. Therefore, it is valuable to offer complete implementations of these programs so that students can inspect their code in a familiar language and experiment with running them.

In the Fall 2012 version of the KBAI course, Goel offered PAIP in Python as a resource for learning AI programming. However, because the course does not teach programming, we did not track the use of PAIP in Python. More recently, AAAI included PAIP in Python on its AITopics web portal (http://aitopics.org/link/paip-python).

We believe that providing working implementations of the algorithms studied in an introductory AI course and examples of their use can help students master the concepts by providing a testbed for experimentation. The PAIP in Python project is such a resource: it is open-source, easily obtained, and written in a language with which students are probably already familiar. Additionally, the project can be integrated into an existing course by providing a platform for course projects; students can demonstrate concept mastery by using or extending the existing programs in various ways, such as:

- Use Emycin to create a new expert system.
- Write a game that uses pathfinding for an AI player.
- Write a Checkers playing program by adapting the program for playing Othello.
- Write a logic program to solve a crime-scene mystery.
- Extend the Prolog interpreter by adding new features.

We look forward to other instructors using PAIP in Python in their own courses.

References

Norvig, P. (1992) *Paradigms of AI Programming: Case Studies in Common Lisp*. Morgan Kauffman.

Abelson, H., Sussman, G, & Sussman, J. (1984) *Structure and Interpretation of Computer Programs*. MIT Press.