Presenting AI to Non PhD-Bound Students

Frank Klassner
Computer Science Department
University of Massachusetts
Amherst, MA 01003
klassner@cs.umass.edu

Introduction

Graduate students are often introduced to AI as a collection of research problems from which they will select a dissertation topic and, ultimately, a research career. After they earn their PhD's and become faculty at computer science departments, this view of AI serves them well in attracting their own research assistants. AI curricula for PhD students, therefore, tend to be based on this view of AI as a field of future research promise. However, I argue that this view is not an appropriate basis for an AI curriculum for non PhD-bound students. Considering that this category represents the large majority of computer science students, it is important that undergraduate AI curricula take these students' needs into account. Having taught three undergraduate computer science courses as a research-oriented AI PhD student, I want to address two issues that AI faculty must face in establishing curricula for non PhD-bound computer science students. Using a curriculum sketch this paper discusses (1) how to present AI as a field with practical value and concrete progress as well as research promise and (2) how to provide and encourage links between AI and computer science curricula.

Curriculum Philosophy

Too often, the term "AI" has been applied to a research problem's murkiness or perceived difficulty, leading to the common situation where people consider a problem as "AI" only until a well-understood solution is developed [Shank 1991]. Introductory textbooks and courses designed with the "future research promise" view perpetuate this attitude by presenting AI as a set of fields independently engaged in making computers do various tasks that humans could do well, often for no other purpose than for showing that computers *could* do them.

The curriculum in this paper champions a practical view of AI as a field with the dual goals of making the human interface with computers as anthropocentric as necessary and the computer interface with the environment as complete as necessary. Many advanced undergraduate AI topics such as natural language processing, machine vision, speech recognition, and robot control can easily be motivated with this definition. More importantly, the view makes it possible to present traditional AI topics (i.e. search, planning, logic, knowledge representation, and probabilistic reasoning) along with advanced topics from other fields such as signal processing and cognitive modeling as *tools* that achieve AI's goals to different degrees. The definition deliberately leaves the definitions of "anthropocentricity" and "completeness" to the performance requirements of the AI application at hand. By recognizing that different applications can require interfaces of varying capabilities, this view encourages students to see various AI techniques as components that can be used depending on how powerful each interface has to be. Conversely, it discourages students from dismissing AI techniques simply because they are not universally applicable.

In solving the presentation and motivation problems, the curriculum's AI definition does not restrict the field. It remains respectful of AI's strong research component when we consider research to represent attempts at achieving the dual goals when "necessary" is extended to "possible." The definition does not reduce the field to traditional interface design, since its conception of "interface" goes far beyond the issues of program display layouts. The definition also avoids any commitment to *human* intelligence by acknowledging that we often want computers to interact with the environment or to solve problems differently than humans can. For example, we may want a computer to monitor the environment with infrared sensors or to verify an assertion with resolution theorem proving.

As noted earlier, not all college students are bound for graduate work; however, AI is often portrayed to undergraduates as a field whose only available jobs require doctorates. If we want to encourage students to enroll in AI courses and apply AI techniques to their future work, the field of AI must be portrayed as a source of solutions to practical problems arising in the interaction between humans and computers. Although industry might not currently be sanguine about AI research per se, it does look favorably on many traditional skills used as tools in achieving the performanceoriented AI interface goals: statistical reasoning, logic analysis, signal processing, etc. Students who employ AI techniques or use traditional skills in nontraditional ways to solve practical problems will make AI-flavored solutions more familiar and attractive to industry.

In an AI curriculum emphasizing practicality, programming must be an important component. Introductory courses without programming can leave students (especially those with little experience in programming nontrivial projects) with a picture of AI as "magic programming." Confrontations with combinatorially explosive search problems in programming assignments make theoretical results in search complexity analysis more immediate to undergraduates who are often still developing their own theoretical reasoning capabilities. While many undergraduates may not be interested initially in pursuing AI to the graduate level, programming assignments that bring some of the experimental fun of AI down from the graduate level can provide interesting enticements.

Curriculum Content

I propose an AI curriculum with two courses and a LISP programming laboratory. The first course is an introductory course offered in conjunction with the programming laboratory to second-semester juniors. The course would include a series of assignments following a progression from evaluating the behavior of complete software packages such as Macsyma to modifying programs for performance improvements to building a final project from scratch. The initial behavior-evaluation assignments and progression in assignment complexity are important since the programming laboratory will require some lead time to orient those students who have had no LISP experience. The second course is an advanced-topics course offered to first-semester seniors. It would rotate among topics such as machine learning, natural language processing, and machine perception.

The debate over using LISP versus some other language for assignments is too involved to present here completely. The curriculum outlined in this paper favors LISP because (1) its use would provide extended experience in the functional programming paradigm, (2) its support for list and type-inspecific symbols eliminates the drudgery of re-implementing the support in another language, and (3) its support for rapid prototyping makes experimental program modification assignments more feasible.

After an introduction to AI problems and evaluation, the first course would present the basic AI topics of search, planning, and knowledge representation during the first half of the semester and then repeat this material in the second half with emphasis on the use of probability, other evidential reasoning frameworks, and logic techniques to control search and planning. Continuing the curriculum's emphasis on applied AI tools, the advanced course's first month would be a tutorial in the advanced-topic's applied skills (e.g. applied signal processing for machine perception, linear algebra for neural nets). In both courses, the curriculum's AI definition gives the instructor the means to present historically significant programs as examples of techniques for solving AI-related problems under various limiting circumstances (e.g. planning where subgoals don't clobber other subgoals, or image processing where scenes are high-contrast), rather than as failed attempts at creating intelligence.

The advanced-topics course is intended to give students the opportunity to apply basic AI techniques extensively in a real problem domain and to glimpse the domain's research boundaries that might entice them into graduate school. While the introductory course would use problem instances from various AI subdisciplines such as natural language processing or game-playing to motivate homework assignments, the advanced course's dedication to one or two topics should ensure that students leave with a feeling that AI's subdisciplines are substantial. The topic rotation not only allows instructors to periodically refresh their acquaintance with AI fields outside their immediate interests, but also prevents the curriculum from degenerating into a course sequence dealing exclusively with the instructor's research forte.

Links with the Local Academic Environment

At the undergraduate level it is important to ground AI within the computer science discipline. Computers and computation are, after all, central to the field's attempts at implementing entities that interact "intelligently" with humans and the environment. An undergraduate AI curriculum should therefore unhesitatingly point out and explain similarities between techniques and tools used in AI and various computer science fields. Such a grounding combines with programming assignments to prevent students from developing an "AI as magic programming" view. The curriculum sketched here offers several possibilities for substantively linking AI with a computer science department's existing course of studies.

In many programming languages (PL) courses, the functional programming paradigm is often presented via a two- or three-week LISP tutorial. The LISP programming laboratory provides the opportunity for a more comprehensive presentation of the paradigm. If the department schedule does not permit an extra programming laboratory, the AI curriculum could rely on an extended treatment of LISP in the local PL course. In either case, the AI curriculum and the PL course can reinforce each other.

Computer science undergraduate students often receive their only exposure to statistical interpretation and probability theory in a third-year mathematics course that focuses on these topics' formal bases. By presenting evidential-reasoning methods from Bayesian theory and Dempster-Shafer theory as tools for controlling search, the introductory AI course can provide students with an example of how the theory behind probability and statistics can be used practically. If the department schedule requires the probability and statistics course in the fourth year, the department can still profit from the third-year introductory AI course as an applied introduction to the topics' later formal treatment.

The majority of computer science curricula offer a course in data structures and/or complexity analysis in the second or third year. The introductory AI course, through its emphasis on search, can serve as an excellent environment for reinforcing students' understanding of graph-traversal algorithms. Lectures on the requirements of maintaining various knowledge representations can give students a more intuitive grasp of complexity analysis techniques.

Finally, and perhaps most importantly, the AI curriculum can serve as a bridge between central computer science topics and outside fields. During questions after his "Paper Tiger in a Cage" invited talk at the 1993 AAAI conference, Edward Feigenbaum lamented the restricted range of computer applications traditionally presented to undergraduate computer science students. Because the advanced-topics course will present practical knowledge from fields such as biology, electrical engineering, and psychology, it can broaden students' capacity for applying their skills in "nontraditional" fields. In 1991 the ACM/IEEE-CS Joint Curriculum Task Force [Turner 1991] recommended that undergraduate computer science programs provide approximately 11 lecture hours on social, ethical, and professional issues. The advanced-topics course can provide an excellent forum for the AI instructor and possibly a guest lecturer from the local philosophy department to address these issues.

Conclusion

Faced with the less research-oriented futures of typical undergraduate students, AI faculty must present their field as being practical and progressive in addition to being full of research promise. This problem takes on special significance for AI's future when one considers that while most undergraduate students probably will not extend AI's research in PhD dissertations, they all have the potential to extend AI's recognition in industry.

The curriculum sketched in this paper tries to solve the practicality problem by (1) presenting existing AI techniques as tools for achieving various levels of user/computer and computer/environment interaction and (2) relating AI to other computer science topics. At the same time, the curriculum's advanced-topics course remains respectful of AI's strong research component by introducing students on a rotating basis to selected subfields and their open issues.

Acknowledgments

I am grateful to Scott D. Anderson and Bruce A. Draper for discussions that clarified my philosophy for teaching AI.

References

[Shank 1991] Shank, R. C., "Where's the AI?" *AI Magazine*, vol. 12, no. 4, pp. 38–49, Winter 1991.

[Turner 1991] Turner, A J., "Computing Curricula 1991," *Communications of the ACM*, vol. 34, no. 6, pp. 69–84, June 1991.