Building Knowledge-Based Systems for Public Use: The Intelligent Reference Systems Project at the University of Houston Libraries

Charles W. Bailey, Jr.

Preprint: 12/16/1988

Introduction

The Intelligent Reference Systems Project was established on January 20, 1988 by the Director of the University of Houston Libraries, Robin Downes, to explore the application of expert systems technology to the reference function. The first goal of the Project was to produce, by the end of the summer, a working prototype of an expert system to assist library users in selecting appropriate indexes and abstracts to meet their information needs, and to advise the director as to whether a production version of this prototype system should be created. The Project's initial team members were Judy Myers (chair), Jeff Fadell, Sandy Maxfield, and Tom Wilson. After Sandy Maxfield assumed a new post at MIT in May, I joined the Project team.

The Project team felt that an iterative prototyping strategy was the best way to develop an expert system. By constructing a series of prototypes, the expert system developer can quickly validate and refine the overall system design, the contents of the knowledge base, and the user interface. Critical problems can be detected and rectified before extensive programming has occurred. Walters and Nielsen provide a good overview of the prototyping approach to expert system development.[1]

Taking its charge to heart, the Project team produced a functional specification prototype and three expert system prototypes by August 31, 1988. The functional specification prototype was an operational mock-up of the system that did not employ expert system technology. The expert system prototypes were working expert systems that represented rough drafts of the expert system we wanted to create.

The Project team recommended that development be continued on the system, and the Director endorsed this decision. Currently, the production version of the system is being programmed by the Project team.

Initial Development Activities

The first focus of the Project team was on the system's user interface. The Project team believed that the system should embody the following user interface design philosophy: (1) the system should present a defined set of choices to the user, rather than accept free-text user input; (2) the system should preview to the user what the results of a choice would be; (3) the system should display to the user that individual's past choices; and (4) the system should allow the user to easily back up to prior choices.

Several general system design principles were also determined. These were: (1) the system should have data-independence (i.e., the data should not be embodied in the source code but, rather, should be kept in easily maintained external files); (2) the system should lay the foundation for future work with other types of reference sources and systems; and (3) the system should be transportable to other libraries. The Project team also tackled the difficult issue of what the primary orientation of the system interface should be: type of material sought, subject of the research topic, depth and sophistication of information needed, etc. After much discussion, the Project team came to the decision that, although no one approach was indisputably superior, a subject approach best reflected

user needs for this particular system; however, this strategy would have to augmented with knowledge about other user information criteria.

Functional Specification Prototype

With these principles in mind, Jeff Fadell created a functional specification prototype for the system using Dan Bricklin's Demo Program II. Since it has very sophisticated graphic capabilities, the Demo program turned out to be an excellent choice for building this prototype. The prototype helped to clarify the Project team's vision of the system, and, as a result of team discussions, the prototype was successively refined until it represented the desired specification for the system.

The functional specification prototype had a graphic, menu-driven user interface. The user was initially presented with broad subject categories shown in boxes on the screen. The user was asked to pick one category. As the user highlighted different boxes using the arrow keys, a box at the bottom of the screen previewed the subject categories that were below the chosen category in the subject category hierarchy. When the user selected a broad category with the Return key, the next screen displayed the lower-level subject categories in boxes, and the user was asked to choose one. One box also allowed the user to pick the broader subject category that had just been viewed. The subject category hierarchy could be of any depth. After selecting a subject category, the user was given the option of qualifying the search with an aspect of the subject, such as the business aspect of art. Once the user had picked the appropriate subject aspect box, the user was asked whether the index should be selective or comprehensive in its coverage of the subject. Whenever the user made a choice, that choice was shown in a numbered box in the upper-left hand corner of the screen. These numbered boxes provided a visual representation of the user's system interaction history, and provided a way for the user to easily back up to any prior choice by simply typing its number. Entering a "Z" at any point allowed the user to start over.

VP-Expert Prototype

Thanks to Jeff's hard work on the functional specification prototype, the Project team knew what the system's user interface should be. What was unclear was whether it would be possible to construct an expert system that had these characteristics. Before attempting to develop a full-featured expert system prototype, the Project team needed to validate their design using an expert system tool and gain experience with expert system technology. It was felt that an inexpensive rule-based expert system shell would be adequate for this purpose.

The Project acquired VP-Expert, which is a very price-competitive expert system shell. Using VP-Expert's inductive capabilities, Jeff defined a two-dimensional table that specified the logical relationships of system data elements and, after VP-Expert translated this table into a rule knowledge base, programmed a user interface for accessing this knowledge base.

The VP-Expert prototype demonstrated that a operational expert system based on Project's system design could be easily constructed from a low-cost expert system shell. As expected, VP-Expert's capabilities were too limited to create the user interface that the Project team desired.

Frame-Based Knowledge Representation

After gaining some experience with VP-Expert, the Project team explored in depth different knowledge representation structures that could be used to construct the expert system prototype. Based on this investigation, the Project team decided that frames offered the best alternative for the prototype. The CoalSORT prototype developed by Monarch and Carbonell demonstrated that a frame-based knowledge representation scheme was useful

for creating an intelligent interface to a bibliographic retrieval system.[2] This research encouraged us to experiment with a frame-oriented approach in our application.

Frames are used to store knowledge about objects.[3] Each frame has a unique name. A frame is composed of a variable number of slots, each of which is used to store information about one attribute of an object. Frames are organized in a hierarchical fashion, where each frame has one or more parents. A frame can inherit slots and values for those slots from frames that are higher up in the hierarchy. Frames can contain built-in rules that execute when a value for a slot is added or changed. Frames can also be used in conjunction with external rules, which can access and manipulate information stored in frames.

Frames appeared to be a ideal knowledge representation structure for meeting the Project's needs. Hierarchical knowledge structures, such as the Project's subject category scheme, could be easily represented. Objects that had a variable number of attributes, such as indexes classified under different numbers of subject categories, could also be easily represented. Using frames, information about an index could be entered once. If information about an index was represented by rules, this information may need to be entered numerous times. For example, if an index were classified under five subject categories and each category had a rule like "IF Subject = Education THEN Index = ERIC", the index information would need to be entered and maintained in at least five rules.

Intelligence/Compiler Prototype

Unfortunately, the majority of low- or moderate-cost expert system shells the Project team investigated did not support frames. However, the Project team was able to identify and acquire a powerful expert system shell, the Intelligence/Compiler from IntelligenceWare, which was available at moderate cost.

The Project team envisioned two primary frame knowledge bases: one for reference works and one for the subjects they covered. The Project team evolved provisional structures for these two types of frames, and team members collected data on these frame types for a representative sample of indexes and abstracts in the library's collection. For the reference works knowledge base, the Project team devised a fairly complex frame structure that embodied their judgments about reference work characteristics, such as ease of use, and supplementary information about reference works that would not normally be found in bibliographic records, such as the types of information that the work would help the user find (e.g., articles, book reviews, etc.).

Based on the information that the Project team collected, Judy Myers and I began to try to build the expert system prototype using the Intelligence/Compiler. Although we were able to construct the appropriate frame knowledge bases, we ran into problems when we tried to retrieve and manipulate information from the knowledge bases. There were four basic problems. First, we found that doing logic programming using the Intelligence/Compiler's Prolog-like language was substantially different from doing conventional programming in a procedural language like Pascal. To be an effective programmer in this environment, you must both unlearn procedural programming habits to take full advantage of the power of logic programming and learn how to make logic programming procedural when required. Second, the program's inherent graphics capabilities were too limited to construct the user interface that the Project team wanted. Third, we considered the user documentation to be inadequate for our purposes, and, since the program is not widely used, there were no other materials that described the specifics of programming the Intelligence/Compiler in depth. Finally, we felt that user support from the company did not meet our needs.

At this stage, it was clear that, in spite of the problems I have just discussed, the Intelligence/Compiler was a very powerful expert system shell and, if we could just get a better handle on how it worked by studying the large number of example programs on disk, we would probably be able to construct an operational prototype using this package. However, we also felt that the user interface was very important, and, since we were

unlikely to create this interface using the Intelligence/Compiler, we should again assess our development options.

Prolog Prototype

We had been drawn to expert system shells because they reduced the amount of programming that we would need to do; however, we realized that there was a trade-off between level of effort and level of control. A logic programming language like Prolog or LISP would give us greater control over the system we were creating than an expert system shell would. Since Turbo Prolog offered speed, power, and sophisticated graphics capabilities at a low cost, it was an attractive development tool; however, we were uncertain how long it would take to program the system in Turbo Prolog. We decided to try a dual track development approach: I would continue to work with the Intelligence/Compiler and Judy would work with Turbo Prolog.

Evaluation of Intelligence/Compiler and Prolog Prototypes

Both expert system programmers invested a considerable amount of energy in creating his or her prototype, and both had to overcome a number of obstacles inherent in attempting to create a relatively complex expert system. Nonetheless, both prototypes were delivered within the scheduled time frame, although neither prototype had all the features desired by the Project team.

The Intelligence/Compiler prototype fully embodied the independent frame knowledge base that was desired, permitted the user to descend through the levels of the subject hierarchy before retrieving information, limited a search by index comprehensiveness, and it ordered retrieved indexes by their ease of use. This last feature represented an addition to the specification that the Project team had wanted. The prototype had a more attractive user interface than I initially thought possible, but it did not meet the Project's goals. Nor did it limit searches according to aspect of subject (e.g, the business aspect of art).

The Prolog prototype provided a close approximation of the user interface we desired; however, it did not fully limit searches according to specifications, permit descending through the subject hierarchy, or embody a fully independent frame structure.

Future Expert System Development Plans

Overall, the Project team felt that the Prolog approach represented the most promising avenue for future development, and the team unanimously endorsed this approach. The next phase of development work will focus on fully implementing search limitations, bringing the system's user interface into full conformity with the prototype specification, moving the frame knowledge bases to ASCII files for easy maintenance, and modularizing and improving the system's code.

After the production version of this system is completed, the Project will pick a subject area, such as business, and develop a second expert system to assist users with identifying all types of reference materials in this subject area that will meet their information needs, such as CD-ROM databases, directories, encyclopedias, handbooks, etc.

References

1. John Walters and Norman R. Nielsen, *Crafting Knowledge-Based Systems: Expert Systems Made Realistic* (New York: John Wiley & Sons, 1988): 109-143.

2. Ira Monarch and Jaime Carbonell, "CoalSORT: A Knowledge-Based Interface," *IEEE Expert* 2 (Spring 1987): 39-53.

3. Kamran Parsaye and Mark Chignell, *Expert Systems for Experts* (New York: John Wiley & Sons, 1988): 161-203.

Citation

Charles W. Bailey, Jr., "Building Knowledge-Based Systems for Public-Use: The Intelligent Reference Systems Project at the University of Houston Libraries," in *Convergence: Proceedings of the Second National Conference of the Library and Information Technology Association*, October 2-6, 1988, ed. Michael Gorman (Chicago: American Library Association, 1990),190-194. http://www.worldcat.org/oclc/869246080

Bailey, Charles W., Jr. "Building Knowledge-Based Systems for Public-Use: The Intelligent Reference Systems Project at the University of Houston Libraries." In *Convergence: Proceedings of the Second National Conference of the Library and Information Technology Association*, October 2-6, 1988, ed. Michael Gorman, 190-194. Chicago: American Library Association, 1990. http://www.worldcat.org/oclc/869246080