https://ntrs.nasa.gov/search.jsp?R=19880007019 2018-03-19T21:30:55+00:00Z

N88-16401

INTERACTIVE KNOWLEDGE ACQUISITION TOOLS

Martin J. Dudziak and Jerald L. Feinstein
ICF/Phase Linear Systems, Inc.
9300 Lee Highway
Fairfax VA 22031-1207
(703) 934-3800

Abstract: This paper discusses the problems of designing practical tools to aid the knowledge engineer and general applications used in performing knowledge acquisition tasks. At issue for the knowledge engineer are several problems and misconceptions of knowledge engineering and knowledge-based systems development. The authors propose a strategy for removing some of those problems, presenting a particular approach we have developed for one class of knowledge acquisition problem characterized by situations where acquisition and transformation of domain expertise are often the bottleneck in systems development. The focus at ICF/Phase Linear has been upon the processing of text-based source materials through a software tool designed in-house, the Knowledge Acquisition Module (KAM). The authors go on to discuss how the tool and the underlying software engineering principles can be extended to provide a flexible set of tools that allow the application specialist to build highly-customized knowledge-based applications.

Introduction

There are some misconceptions or misrepresentations regarding what knowledge engineering can or should do. These confusions result in a failure to make the best use of computer technology and artificial intelligence-based techniques for building knowledge-based systems that are reliable and effective for real-world applications. The knowledge engineering process is typically characterized as follows:

(1) There is a body of expert knowledge "out there" which is in the minds of certain experts (or what they have produced -documents, automated systems, etc.).

(2) This knowledge can be codified or summarized into a formal representation which can then be used by an automated system.

(3) The knowledge engineer must "obtain" that body of knowledge and transform it into the ideal type of symbolic representation - discovering the ideal formalism is a goal that must be attained.

(4) The symbolic representation must be implemented into an expert system where there is a mapping of the symbolic representation into some form of code.

(5) Once built the expert system should perform its assigned tasks in a manner that is predictably similar to the way a human expert would carry out those tasks.

A fundamental misconception is that a comprehensive body of knowledge exists in the first place which can be codified into a formal representation. There is a tendency to think of knowledge as objects, facts as being entities that can be bounded and enclosed within the descriptive framework of a given type of formalism. There is also a tendency to think of the mapping problem (expert knowledge into symbolic form) as a task that has a singular and finite answer. However, putting automated systems aside and considering for a moment only human exchanges of information and learning, it is clear that acquisition and transfer of knowledge is not a linear sequence or an early codifiable phenomenon. The expert-novice interchange is highly iterative and interpolative. By this it is meant that the exchanges are more like conversations rather than data transfers as we normally think of them. [1] As with conversations, the implied background knowledge of both persons in the exchange becomes highly significant for the correct interpretation of what is spoken by both (all) participants.

There must be a high level of dialogue, particularly interrogation in both directions between expert and novice. This "handshaking", as it were, is what enables both participants to know that the other is understanding what is being communicated. Such questioning enables the novice to make clear what is understood and what is unclear and what is his or her context of understanding; it also empowers the expert with knowledge about the communication process so that he can emphasize or clarify certain facts, rules, and relationships. In a knowledge acquisition activity, particularly between engineer and expert, frequent questioning and clarification is the key to making sure that both are "speaking the same language." Of course, this often leads to a increased volume of written and verbal material to be analyzed and deciphered; thus the need for automating parts of those processes.

The knowledge engineer acts as both the go-between for an expert and an automated system and also as the designer of that computer-based product and must recognize this dual nature to this work. The knowledge engineer must take the lead in focusing the knowledge acquisition process so that it serves to not only provide substantive expressions of the expert's knowledge but information that will help in designing the most appropriate system structures for using that knowledge in the automated application. The knowledge engineer is responsible in a way unlike the typical apprentice to the expert in that what is relevant or useable information must be defined. Also the system design must be modified in response to new expert information that is gained through the interviews, dialogues and other acquisition activities.

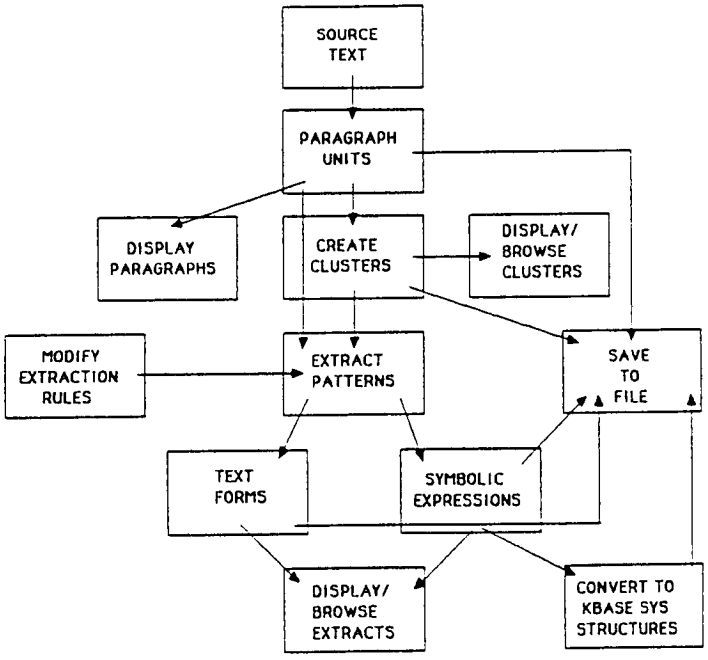
It is critical to keep the knowledge engineer active at every level of the acquisition process, from interfacing with the experts to formulating a computer-based representation for what was obtained. This provides the broad-context element of control which can take into account not only the expert source material currently being processed (e.g., codified) but also the knowledge which may not be coded into any automated system and which may not at the moment seem directly relevant but which can become relevant in the future. But if the knowledge engineer is to be actively involved in the entire acquisition-representation process, the tools that will help carry the load and to perform those tasks in a reasonable period of time must be available.

In brief, the knowledge engineer needs tools that can expedite these tasks but not perform them without active participation and control. These tools must enhance productivity and correctness without adding to the work load, rather than tools which replace entire segments of the knowledge acquisition process. If the knowledge engineer is taken out of the loop, so to speak, then the opportunity to bring in the broad-context of both acquired or implicit expert knowledge as well as general common-sense knowledge is reduced. Moreover, the task of identifying situations where a highly-automated module has generated something in need of correction or has omitted something, and the task of making those corrections or modifications "after the fact", may be so time-consuming and tedious that the value of the initial automation process is negated.

One insufficiently-addressed aspect of the knowledge engineering process which can be significantly improved and which has been the focus of ICF/Phase Linear automation efforts concerns the extraction of information imbedded in free-format source materials (e.g., texts, transcripts) and the transformation of such knowledge into useable formal representations. The latter may be in the syntax of the knowledge engineer's application system under development or in some intermediate form which the knowledge engineer may have adopted. There are problems not only in transforming loosely-formed knowledge into a codified symbolic representation but also in handling voluminous and diverse-format database records, text, interview transcripts, and other digitizable material. The problems in acquiring the knowledge properly also affect the selection and formalization of a sufficiently robust representational scheme to be used in the actual expert system, planner, scheduler or other application. The knowledge engineer needs to have fluent and easy access to the breadth and depth of relevant source material to effectively design data structures that will store facts, rules, relations and to design or select the reasoning mechanisms that will manipulate the knowledge bases.

In its project management and consulting work, ICF/Phase Linear staff have frequently found situations where the volume of interview transcripts, background texts, and reference materials,

particularly manuals and project specification documents, posed the major roadblock to establishing even an elementary knowledge base for an application. This has been particularly true for such engineering applications as autonomous underwater and aerospace vehicle control, mission planning, fault diagnosis and maintenance. The approach ICF/Phase Linear has been developing consists of building relatively simple software tools which to reduce the amount of material which the engineer must handle. Such tools also help to create intermediary data structures that can be directly applied toward the next phases of the knowledge engineering process such as incorporation into relations, facts, and rules in a knowledge base. The goal has been to allow the knowledge engineer to move quickly through large and difficult masses of data and to provide the ability to create new data structures that are more condensed, focused and easily managed, primarily through rapid browsing and editing. The result is that the engineer can be more involved in the full knowledge acquisition process without being overwhelmed by time-consuming operations.



KAM FUNCTIONAL STRUCTURE

KAM or the Knowledge Acquisition Module has been implemented as the kernel tool in a family of such expediter tools. As such, it is a concrete example of how some of the strategies outlined above can be feasibly implemented in a low-cost software package running on low-cost general-purpose hardware such as the PC family of microcomputers. The figure above illustrates the basic KAM functional structure.

In its first phase KAM has been built to handle source files of text data but it can be extended to work with non-text data as

well. Most applications for this type of program will involve text but in a variety of different formats besides standard paragraph-oriented text files. The primary features KAM provides are:

- The ability to rapidly form and browse through clusters or subsets of text arranged according to topics (the latter being specified using strings and keywords but also logical relationships between words, presence of synonyms and morphemes).

- The ability to extract elements of text on the sentential level which match pattern templates (rules that are defaults or established by the user) and to browse through these extracted patterns.

- The reconstruction of extracted text elements into both simple English-like and code-like user-specified representational forms which can be used directly or after editing in application systems.

- The ability to browse quickly among extracts and source text and to make edits to extracts which are automatically reflected in the other representational forms of the extracts.

- The ability to work with multiple source files and extract data sets and to combine data produce from different sources.

- The ability to automatically generate data structures from extracts according to pre-specified syntax rules such that the output data sets can be input into the knowledge bases of various existing applications.

KAM is currently implemented in LISP on a PC/AT. It has been developed as a prototype following an initial proof-of-concept version built on a Symbolics workstation, and further extensions and refinements of the system will be oriented toward deliverability on a variety of hardware besides the PC/AT.

While KAM is designed to be used in a standalone capacity for extracting and transforming text, it can also be incorporated, modular-fashion, into a more comprehensive workstation environment that provides the user with the capability of defining and using ad hoc "knowledge-object" definitions - software structures which specify different classes and types of data that the user may discover a need to use during the knowledge acquisition life cycle. An example of such a definition is one called a TopicDef; it is a frame-like structure that specifies the different rules and functions to be employed by KAM and related applications for determining whether or not a given piece of text should be considered as bearing reference to a topic or not. The TopicDef instructs KAM as to how the user conceives of the topic and how

the system should proceed in its examination of texts in order to judge its relevance or not.

Only brief mention has been made of the fact that there is a broad class of straightforward software tools for the knowledge acquisition process and that like KAM, they can be developed into standalone units or integrated to form a workstation environment. Such an environment is the application user's equivalent of a programmer's development environment as is found on a variety of machines, the most obvious perhaps being UNIX on conventional hardware and the Symbolics LISP machines. By providing more fluidity and convenience at the workstation, the knowledge engineer can address the acquisition problem and increase productivity in much the same fashion as the AI programmer can more effectively grapple with program design and prototyping issues through well-established features like incremental compilation, run-time debugging and editor-level evaluations. The behaviors of programming and knowledge acquisition are not that dissimilar. It seems appropriate to expand the tools which have proven successful in the programming arena toward time-consuming, productivity-draining problems in critical application areas like knowledge acquisition.

[1] A source for much of the theoretical foundation for concepts expressed in this paper is: Winograd, T. & Flores, F., "Understanding Computers and Cognition", Addison-Wesley, Reading MA, 1987