PORTABILITY IN THE JANUS NATURAL LANGUAGE INTERFACE[1]

Ralph M. Weischedel, Robert J. Bobrow, Damaris Ayuso, Lance Ramshaw
BBN Systems and Technologies Corporation
10 Moulton Street
Cambridge, MA 02138

ABSTRACT

Although natural language technology has achieved a high degree of domain independence through separating domain-independent modules from domain-dependent knowledge bases, portability, as measured by effort to move from one application to another, is still a problem. Here we describe a knowledge acquisition tool (KNACQ) that has sharply decreased our effort in building knowledge bases. The knowledge bases acquired with KNACQ are used by both the understanding components and the generation components of Janus.

INTRODUCTION: MOTIVATION

Portability is measurable by the person-effort expended to achieve a pre-specified degree of coverage, given an application program. Factoring an NL system into domain-dependent and domain-independent modules is now part of the state of the art; therefore, the challenge in portability is reducing the effort needed to create domain-dependent modules. For us, those are the domain-dependent knowledge bases, e.g., lexical syntax, lexical semantics, domain models, and transformations specific to the target application system.

Our experience in installing our natural language interface as part of DARPA's Fleet Command Center Battle Management Program (FCCBMP) illustrates the kind of portability needed if NL applications (or products) are to become widespread. We demonstrated broad linguistic coverage across 40 fields of a large Oracle database, the Integrated Data Base (IDB), in August 1986. A conclusion was that the state of the art in understanding was adequate. However, the time and cost needed to cover all 400 fields of the IDB in 1986 and the more than 850 fields today would have been prohibitive without a breakthrough in knowledge acquisition and maintenance tools.

We have developed a suite of tools to greatly increase our productivity in porting BBN's Janus NL understanding and generation system to new domains. KREME [Abrett, 1987] enables creating, browsing, and maintaining of taxonomic knowledge bases. IRACQ [Ayuso, 1987] supports learning lexical semantics from examples with only one unknown word. Both of those tools were used in preparing the FCCBMP demonstration in 1986. What was missing was a way to rapidly infer the knowledge bases for the overwhelming majority of words used in accessing fields. Then one could bootstrap using IRACQ to acquire more complex lexical items.

We have developed and used such a tool called KNACQ (for KNowledge ACQuisition). *The efficiency we have experienced results from (1) identifying regularities in expression corresponding to domain model structures and (2) requiring little information from the user to identify expressions corresponding to those regularities.*

---

[1] This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contracts N00014-85-C-0079 and N00014-85-C-0016. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

WHAT KNACQ DOES

KNACQ assumes that a taxonomic model of the domain exists, such as that typical in many expert systems, and assumes that it is encoded in an axiomatizable subset of KREME [Brachman, 1985]. At this point we have built translators for transforming KEE taxonomies and PCL hierarchies into KREME structures.[2] The browsing facilities, graphical views, and consistency checker of KREME are therefore at the disposal of the knowledge base administrator or knowledge engineer when using KNACQ.

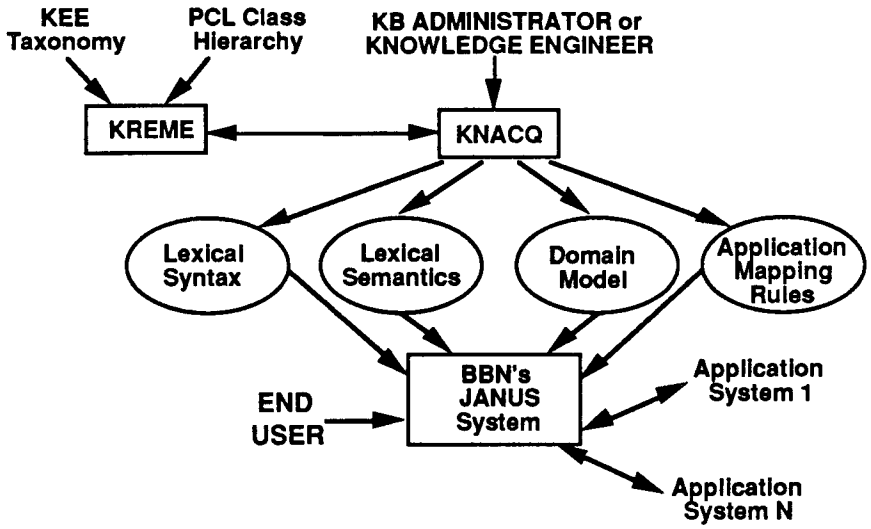

Figure 1: Role of KNACQ

Using KREME, users may select any concept or role for processing. KNACQ presents the user with a few questions and menus to elicit the English expressions used to refer to that concept or role. From the user's answers, KNACQ creates simple structures which together with powerful general rules allow understanding of a wide range of expressions.

To illustrate the kinds of information that must be acquired consider the examples in Figure 2. To handle these one would have to acquire information on lexical syntax, lexical semantics, and mapping to expert system structure for all words not in the domain-independent dictionary. For purposes of this exposition, assume that the following words, *vessel, speed, Vinson, CROVL, C3*, and *deploy* are to be defined. *A vessel has a speed of 20 knots* or *a vessel's speed is 20 knots* would be understood from domain-independent semantic rules regarding *have* and *be*, once lexical information for *vessel* and *speed* is acquired. In acquiring the definitions of *vessel* and *speed*, the system should infer interpretations for phrases such as *the speed of a vessel, the vessel's speed*, and *the vessel speed.*

The vessel speed of Vinson
The vessel's speed is 5 knots
Its speed
Which vessels are deployed C3?
The vessels with speed above 20 knots
Vinson has speed less than 20 knots
Which vessels have a CROVL of C3?

Figure 2

[2] Of course, it is not the case that every piece of knowledge statable in KEE taxonomies and PCL hierarchies has a correlate in the axiomatizable subset of KREME. We do not guarantee that the NL interface will understand English expressions corresponding to anything falling outside of the axiomatizable subset.

Given the current implementation, the required knowledge for the words *vessel*, *speed*, and *CROVL* is most efficiently acquired using KNACQ; names of instances of classes, such as *Vinson* and *C3* are automatically inferred from instances in the expert system taxonomy; and knowledge about *deploy* and its derivatives would be acquired via IRACQ. That is, *we recommend using IRACQ for the diverse, complex patterns of syntax and semantics arising from verbs by providing examples of the verbs' usage, while using KNACQ for efficient acquisition of the more regular noun phrase information (excluding verb-based constructions).*

KNACQ FUNCTIONALITY

Five cases are currently handled: one associated with concepts (or frames), two associated with binary relations (or slots), and two for adjectives. In each case, one selects a concept or binary relation (e.g., using the KREME browser) to provide lexicalizations for that domain entity.

CONCEPTS OR CLASSES

The association of English descriptions with concepts is the simplest case. It is fundamental knowledge about unmodified head nouns or frozen nominal compounds from which we can build more powerful examples. KNACQ must acquire one or more phrases for a given class, and their declension, if irregular. For the concept CARRIER of Figure 3, we provide KNACQ with the phrases *carrier* and *aircraft carrier*, which can be treated as a frozen nominal compound. Since both are declined regularly, no further information is required. One can provide *surface vessel* for SURFACE-VESSEL in Figure 3, but that would not allow compositions, such as *Count the surface and subsurface vessels*. Rather, one should define *surface* and *subsurface* as non-comparative adjectives (Section 3.4) modifying phrases corresponding to VESSEL in order to define phrases for the concepts SURFACE-VESSEL and SUBSURFACE-VESSEL.

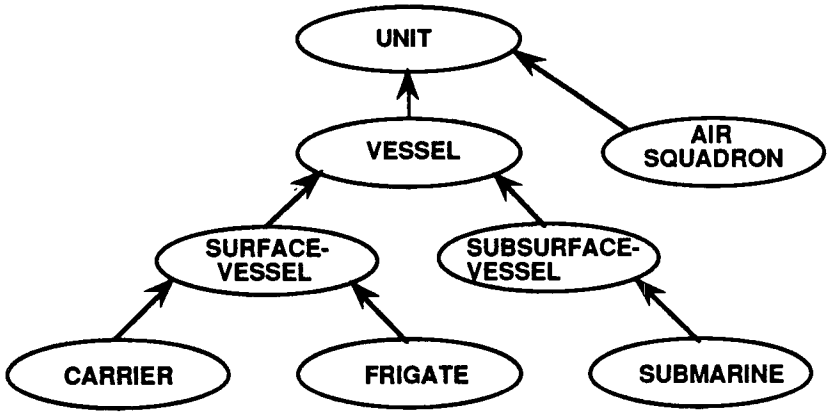

Figure 3: Simple Class Hierarchy

ATTRIBUTES

Attributes are binary relations on classes that can be phrased as *the <relation> of a <class>*. For instance, suppose CURRENT-SPEED is a binary relation relating VESSEL to SPEED, a subclass of ONE-D-MEASUREMENT. An attribute treatment is the most appropriate, for *the speed of a vessel* makes perfect sense. KNACQ asks the user for one or more English phrases associated with this functional role; the user response in this case is *speed*. That

answer is sufficient to enable the system to understand the kernel noun-phrases listed in Figure 4. Since ONE-D-MEASUREMENT is the range of the relation, the software knows that statistical operations such as average and maximum apply to speed. The lexical information inferred is used compositionally with the syntactic rules, domain independent semantic rules, and other lexical semantic rules. Therefore, the generative capacity of the lexical semantic and syntactic information is linguistically very great, as one would require. A small subset of the examples illustrating this without introducing new domain-specific lexical items appears in Figure 4. *It is this compositionality and the domain independent rules that provide the utility of KNACQ.*

KERNEL NOUN PHRASES

the speed of a vessel *the vessel's speed* *the vessel speed*

COMPOSITIONALLY WITH OTHER LEXICAL SEMANTICS, SYNTACTIC RULES, AND SEMANTIC RULES

Which vessels have speed above 20 knots
The carriers with speed above 20 knots
The vessels with a speed of 20 knots
Vinson has speed less than 20 knots
Eisenhower has Vinson's speed
Carriers with speed 20 knots
The vessel's speed is 5 knots
Vinson has speed 20 knots
Which vessels have speeds
The vessel speed of Vinson
Their average speeds
Their greatest speed
Vinson has speed 1
Its speed

Figure 4

CASEFRAME RULES

Some lexicalizations of roles do not fall within the attribute category. For these, a more general class of regularities is captured by the notion of caseframe rules. Suppose we have a role UNIT-OF, relating CASUALTY-REPORT (casrep) and MILITARY-UNIT. Besides asking about the unit of a casrep (the attribute use), a user will want to ask about the casreps on a unit (the inverse direction)—this is one case where caseframe rules are needed. KNACQ asks the user which subset of the following six patterns in Figure 5 are appropriate plus the prepositions appropriate.

1. <CASUALTY-REPORT> is <PREP> <MILITARY-UNIT>
2. <CASUALTY-REPORT> <PREP> <MILITARY-UNIT>
3. <MILITARY-UNIT> <CASUALTY-REPORT>
4. <MILITARY-UNIT> is <PREP> <CASUALTY-REPORT>
5. <MILITARY-UNIT> <PREP> <CASUALTY-REPORT>
6. <CASUALTY-REPORT> <MILITARY-UNIT>

Figure 5: Patterns for the Caseframe Rules

For this example, the user would select patterns (1), (2), and (3) and select *for*, *on*, and *of* as prepositions. Normally, if pattern (1) is valid, pattern (2) will be as well and vice versa. Similarly, if pattern (4) is valid, pattern (5) will normally be also. As a result, the menu items are coupled by default (selecting (1) automatically selects (2) and vice versa), but this default may be simply overridden by selecting either and then deselecting the other. The most frequent examples where one does not have the coupling of those patterns is the preposition *of*.

GRADABLE ADJECTIVES

Certain attribute roles have ranges that may be compared, e.g., numbers or measurements. Adjectives can be given for these roles; assume *fast* is given by the user for the CURRENT-SPEED role or VESSEL discussed earlier. KNACQ can correctly predict the comparative and superlative forms of *fast*. Suppose x and y are instances of

VESSEL. The next information needed is whether *x is faster than y* means *x's speed is greater than y's speed* or *x's speed is less than y's speed.* Optionally, a threshold t can be given such that *x's speed is greater than t* means *x is fast* (for a vessel). Additionally, one can specify antonyms for *fast*, such as *slow*. The information above would enable understanding the expressions in Figure 6.

Is Frederick faster than every carrier?
How fast are the carriers?
Is Vinson fast?
How fast is the fastest carrier?
Which vessels are slower than 20 knots?
Show the fastest vessel.
Is Vinson as fast as Frederick?

Figure 6: Examples after defining *fast*

NON-GRADABLE ADJECTIVES

Of the remaining types of adjectives, some correspond to refining a concept to another named concept in the hierarchy. For instance, *surface* and *subsurface* have that property given the network in Figure 3. In such a case, one must indicate at the general concept, the adjective, any synonyms, and the refined concept.

Others correspond to an arbitrary restriction on a concept having no explicit refined concept in the domain model. Though one could add such a refined concept to the hierarchy, we allow the user to state a logical form to define the adjective as a predicate of one argument.

A case not yet covered in KNACQ is non-gradable adjectives that are predicates of more than one argument. An example in the FCCBMP domain is mission readiness ratings, *M1*, *M2*, *M3*, *M4*, and *M5*. An example is *Enterprise is M2 on anti-air warfare*, where both the vessel and the type of mission are arguments.

EXPERIENCE THUS FAR

There are several things we have learned even in the early stages of KNACQ's development based on porting Janus to CASES, an expert system in DARPA's Fleet Command Center Battle Management Program (FCCBMP). In this use of KNACQ, the original domain model pertinent to the portion requiring a natural language interface consisted of 189 concepts and 398 roles.

First, no restructuring of that domain model was necessary, nor was any deletion required. Second, we found it useful to define some additional concepts and roles. Certain subclasses not critical to the expert system were nevertheless lexically significant. In total, only 123 concepts were added: 53 for classes that were treated as strings in the expert system and 70 domain-independent concepts pertaining to time, space, events, commands, etc. Similarly, 28 roles were added: 24 domain-independent roles and 4 domain-specific roles. In addition, some roles were added to represent role chains that are lexically significant directly. For instance, the DISPLACEMENT of the VESSEL-CLASS of a VESSEL is lexicalizable as *the vessel's displacement.* Starting from a given concept, a procedure exists to run through a subhierarchy checking for role chains of length two to ask the user if any of those are significant enough to have lexical forms. For the example network we needed to add only 5 roles for this purpose. Third, 1093 proper nouns (e.g., ship and port names) were inferred automatically from instances.

As a result, the time required to supply lexical syntax and semantics was much less than we had experienced before developing KNACQ. In two days we were able to provide 563 lexical entries (root forms not counting morphological variants) for 103 concepts and 353 roles. Together with the automatically inferred proper nouns, this was approximately 91% of the domain-dependent vocabulary used for the demonstration. That is about 5-10 times more productivity than we had experienced before with manual means.

RELATED WORK

TEAM [Grosz, 1987] is most directly related, having many similar goals, though focussed on data bases rather than expert systems or knowledge bases. The novel aspects of KNACQ by contrast with TEAM are (1) accepting an expert system domain model as input (KNACQ) contrasted with the mathematically precise semantics of a relational data base (TEAM), and (2) how little information is required of the KNACQ user.

A complementary facility is provided in TELI [Ballard, 1986] and in LIFER [Hendrix, 1978]. KNACQ is meant to be used by the (expert system's) knowledge engineer, who understands the expert system domain model, to define a large portion of the vocabulary, that portion corresponding to simple noun phrase constructions for each concept and role; one uses KNACQ to bootstrap the initially empty domain-dependent lexicon. TELI and LIFER, on the other hand, are meant to let the end user define additional vocabulary in terms of previously defined vocabulary, e.g., *A ship is a vessel*; therefore, those systems assume an extensive vocabulary provided by the system builder. Obviously, providing both kinds of capabilities is highly desirable.

CONCLUSIONS

KNACQ is based on the goal of allowing very rapid, inexpensive definition of a large percentage of the vocabulary necessary in a natural language interface to an expert system. It provides the knowledge engineer with the facilities to browse his/her taxonomic knowledge base, and to state head nouns, nominal compounds, and their non-clausal modifiers for referring to the concepts and roles in the knowledge base. Given that, KNACQ infers the necessary lexical syntactic and lexical semantic knowledge. Furthermore, if appropriate instances in the expert system knowledge base already have names, KNACQ will add proper nouns for those instances to the lexicon.

KNACQ does not cover the inference of complex constructions typical of verbs and their nominalizations. IRACQ [Ayuso, 1987] allows a user to enter examples of usage for acquiring lexical syntax and semantics for complex constructions.

Our experience thús far is that KNACQ has achieved our goals of dramatically reducing the time it takes to define the vocabulary for an expert system interface. It appears to have increased our own productivity several fold. (However, KNACQ has not yet been provided to a knowledge engineer with no knowledge of computational linguistics.)

We believe that the problem of linguistic knowledge acquisition is critical not just as a practical issue regarding widespread availability of natural language interfaces. As our science, technology, and systems become more and more mature, the ante to show progress could involve more and more effort in filling domain-specific knowledge bases. The less effort spent on such knowledge bases, the more effort can be devoted to unsolved problems.

References

Abrett, G. and Burstein, M. The KREME knowledge editing environment. *Int. J. Man-Machine Studies* 27:103–126, 1987.

Ayuso, D.M., Shaked, V., and Weischedel, R.M. An Environment for Acquiring Semantic Information. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 32–40. ACL, 1987.

Ballard, B. and Stumberger, D. Semantic Acquisition in TELI: A Transportable, User-Customized Natural Language Processor. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 20–29. ACL, June, 1986.

Brachman, R.J. and Schmolze, J.G. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science* 9(2), April, 1985.

Grosz, B., Appelt, D.E., Martin, P., and Pereira, F. TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces. *Artificial Intelligence*, Vol. 32, No. 2, May 1987.

Hendrix, G., et. al. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems* 3(2):105–147, 1978.