APPLICATION-SPECIFIC ISSUES IN NATURAL LANGUAGE INTERFACER DEVELOPMENT FOR A DIAGNOSTIC EXPERT SYSTEM

Karen L. Ryan Rebecca Root
Duane Olawsky

Honeywell Inc.
CSDD
1000 Boone Avenue N.
Minneapolis, Minnesota 55427

ABSTRACT

This paper describes a natural language interface developed for an expert system, Page-X. The interface accepts English descriptions of observed symptoms and maps those descriptions to hypotheses used as initial input to the Page-X diagnosis system. The interface describes an application-independent linguistic interface and an application-specific hypothesis identification component.

PROBLEM DESCRIPTION

The natural language interface (NLI) we describe here makes up one part of the user interface to Page-X [STRA85], an expert system for the diagnosis of problems in non-impact page printers. To better understand some of our design decisions, it is useful to see the context in which this system was developed and deployed. Page-X has been in development since 1984 and was first deployed in early 1986. The knowledge base continues to be expanded to handle new printers. Two copies of the system reside in a central location. Field technicians who have the responsibility of serving customer sites are trained to diagnose problems in these printers; however, they often must rely on more expert knowledge. They can access Page-X via modem over a standard ASCII terminal. Page-X performs diagnosis by reasoning from an initial set of symptom hypotheses to a probable cause, typically asking the technician to perform various tests along the way. The task of the user interface is to allow the technician to efficiently state what symptoms he/she has observed, both initially and as a result of the requested tests. Menus were used as the original mode of specifying symptoms, and these form a part of the current interface. However, as the knowledge base grew to thousands of hypotheses, it became unwieldy to input the hypotheses. A keyword interface was added to allow technicians to describe the situation more directly. This interface functioned by matching a predetermined set of keywords in the user's typed free-form English input to the hypotheses' names, which were actually short descriptions. Page-X would then start reasoning from this initial set, but dynamically compose menus to receive input about the results of requested tests. This mode of interaction proved to be very easy to implement. However, the limitations of keyword matching soon became restrictive, e.g.,: (1) it was not possible to be precise enough in the match process; (2) it was difficult to treat synonymous words properly, and (3) in some subdomains (e.g., print quality), the amount of synonymy and near synonymy made exact keyword matching nearly useless.

These factors motivated the developers to replace the keyword part of the interface with a restricted NLI. Subsequent use of dynamically created menus is proceeding as before. This project is currently in its infancy, having started in March of 1986. At present it deals with only a subset of the domain covered by Page-X, namely print quality, and is not as yet complete with respect to this subdomain. However, the results are extremely promising based on a comparison of the possible performance of the keyword interface and the performance of the implemented natural language interface.

SYSTEM OVERVIEW

The Page-X interface can be divided into two major modules: The linguistic interface component (LIC) and the symptom identification component (SIC). The LIC is based almost entirely on an interface developed under the ATOZ project [LARS85]. The SIC had to be specifically developed for the Page-X application and domain.

LINGUISTIC INTERFACE COMPONENT

The LIC consists of several subcomponents that apply various grammars and lexicons to yield a domain- and application-specific interpretation of the input. As in most such NLIs, the components are: (1) a parser that makes use of a grammar and lexicon to produce a constituent structure representation, (2) a logical interpretation component, which uses a set of Montague grammar style rules to produce a "logical form" representation, (3) a lexical translation component, which uses a domain model and rules to translate English terms to domain-model terms, and (4) a logical form translation component, which uses a set of logical form transformation rules to produce a representation adhering to the specific formal language syntax required to interface to the application. A box diagram of the LIC is presented in Figure 1. Adapting the ATOZ LIC to Page-X required writing a new domain model and new lexicons as well as extending each of the three grammars to handle linguistic phenomena not encountered in the original query applications. No software modifications were necessary.

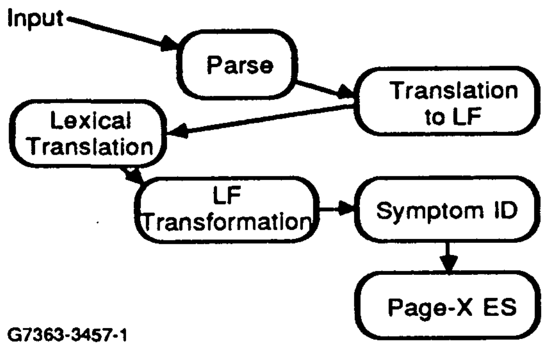

Figure 1. Diagram of the LIC

The output of the LIC cannot serve as direct input to the application program. There will typically be a need for translation from domain predicates to application predicates and/or a matching of input propositions to application propositions. Page-X symptom hypotheses are stated in terms of domain model predicates, so no predicate translation is necessary. However, for reasons that will be explained in detail below, it is necessary to have an explicit component to match the representation of the input to the hypothesis. This is the function of the SIC.

DETAILED INTERFACE TRACE

In this section, we will discuss a detailed trace of one example, showing the overall operation of the system. Figure 2 shows the initial input, the output of each stage of semantic interpretation, and the final set of symptoms identified as closest to the original input. For space considerations, we have suppressed the parser output which is an unremarkable feature-value graph of the constituent structure.

```
> (px-atoz (white line going down page) t)
Parse completed.

Logical Form:
((((WHITE Y1159)) ((^GO E1173 Y1159)
    (VERB-SUBJ E1173 Y159)))
    ((PAGE Y1169) (DOWN E1173 Y1169)))

Lexical and Logical Transformation:
((AFFECTED-OBJECT Y1159)(COLOR-OF
    Y1159 !1176)
    (LIGHT-COLOR !1176)(WHITE
    !1176)(INTENSITY-OF Y1159 !1177)
    (LIGHT !1177)(BAND-LINE Y1159)
    (ORIENTATION-OF Y1159
    !1178)(VERTICAL !1178)
    (NIL)(NIL)(ENTIRE-PAGE
    Y1169)(AFFECTED-OBJECT Y1159)
    (ORIENTATION-OF Y1159
    !1179)(VERTICAL !1179) LOCATION-OF
    Y1159 Y1169)(ENTIRE-PAGE Y1169))

Matcher:
(HYPO-1503.P3-PRINT-HAS-LIGHT-
    VARYING-WIDTH-BANDS
    HYPO-1151.P3-PRINT-HAS-LIGHT-
    VERTICAL-BANDS
```

```
HYPO-1367.P3-PRINT-HAS-
SYMMETRICAL-BANDS
HYPO-251.P3-PRINT-HAS-LIGHT-
BANDS-CONSTANT-WIDTH
HYPO-1374.P3-TONER-STARVATION
HYPO-1371.P3-PRINT-HAS-UNEVEN-
PRINT DENSITY-SIDE-TO-SIDE
HYPO-165.P-3-THE-PRINTED-PAGE-
IS-WASHED-OUT
HYPO-111.P3-PRINT-HAS-VERTICAL-
LINES
HYPO-1370.P3-PRINT-HAS-SLASHES
```

Figure 2. Trace Output

The keyword interface operated on matching the content words found in the hypothesis names. As can be seen from this example, that method would fail to identify a significant number of possible causes.

LINGUISTIC COVERAGE

SYNTACTIC ANALYSIS

Syntactic analysis is performed by means of a unification-based chart parser applied to a combinatory categorial grammar. The parser was developed under the LUCY project at MCC [WITT86]. The grammar employed is the ATOZ grammar, with extensions to handle some aspects of "telegraphic" speech. The scope of this project did not permit a thorough determination of the sublanguage used in this domain and application. However, we were able to study approximately 20 user transcripts and complete one field engineer interview to help us arrive at a working grammar and lexicon. Because the application requires that the user give symptom descriptions, input is almost always in the form of simple declarative clauses and phrases. Some examples are given here:

(1) Fat characters,
(2) No format print,
(3) Overprinting garbage,
(4) Characters smeared down left side of page,
(5) Character too dark,
(6) Characters repeating down page.

As can be seen, structures omitting determiners, copula, whole predicates and whole subjects are often used. Since this appears to be the norm for this application, these are treated as fully grammatical. Logical form rules ensure that they receive the same interpretation as their fully specified counterparts. Because this grammar is an outgrowth of a grammar previously developed for database query, there is also coverage of standard interrogative and imperative structures, though these forms are not present in our user input.

SEMANTIC TRANSLATION

The translation to initial logical form is driven by a Montague-style grammar that pairs constituent structure graphs to expressions of lambda calculus. The translation component itself simply applies these rules recursively and performs formula reduction. An example rule is given here. This would be for translating a noun modified by an adjective:

```
[X: [cat:N
    lf: [rule: ( x (1(x) 2(x)))
    arg1: <1>
    arg2: <2>]]

Y: [cs: [head: 2[cat: N]
    mod: 1[cat: Adj]]]]
```

Graph unification is used to match a graph with the appropriate rule and to supply the arguments to the lambda calculus expression. Since the purpose of this component is to yield a logical formula for every input phrase, the coverage is identical to that of the syntactic grammar. At this point, semantic translation primarily serves to reveal the predicate argument structure of the constituent structure graph. Negation and quantifiers are also handled, although the current versions of the domain model and the SIC do not have mechanisms to deal with these. Because of the limitations of the SIC< certain other semantic distinctions that appear to be important within this domain, such as iterative aspect, tense, and degree, are ignored at this point.

ISSUES

UNGRAMMATICAL INPUT AND PARTIAL INTERPRETATIONS

In describing the grammatical coverage above, we mentioned that certain kinds of nonstandard constructions were treated as fully grammatical. However, it is not possible to take this approach everywhere the input deviates from the standard. For the system to be of use during development, and to take care of idiolectic and random deviance, some treatment of true ungrammaticality is necessary. One approach would be to revert at this point to the simple keyword match. However, even in unparsed sentences, there is frequently enough information available to better identify the intended symptom. The approach we take is to heuristically assemble a partial syntactic analysis made up of three fragments, translate the fragments, and then again apply heuristics to get the best domain-dependent connection between the logical form fragments.

Syntactic phrase assembly is accomplished by using path finding techniques on the chart to discover the best set of chart edges that exhaustively cover the input with no overlaps. An A* technique is employed with path score based on path length and edge length. Short paths are favored over long ones. This ensures that the partial analysis is made up of a few large constituent structure graphs. These general heuristics could be replaced or augmented by heuristics based on more grammar-specific characteristics such as major category or presence of a required subcategory. Translation of logical form and insertion of domain predicates proceed as normal, yielding a set of partially connected formulas with default translations where contexts could not be met.

The SIC, by itself, would not work well at all if given a formula where the variables are not properly shared. It would simply ignore all of the information that was not tied, through variable bindings, to the affected-object predicate. This could be most of the formula. To prevent this, whenever a complete parse is not found, variable bindings are forced in the hypothesis matcher input so that all predicates are tied through shared variables to the affected-object predicate. These forced bindings may, of course, be incorrect, but initial tests indicate that the rather simple-minded approach to the ungrammaticality we employ here still brings noticeable improvement over the straight keyword match.

LEXICAL TRANSLATION ISSUES

Lexical translation is the process of substituting one or more predicates of a domain model for one or more English lexical items. The lexical translation step in the Page-X system is a subpart of the step of translation from English to an intermediate domain model.

Lexical translation as used here is not exactly the same as the problem of mapping from English lexical items to standard database constructs. The problem of translating English lexical items to standard database constructs can be broken into at least two steps: (1) English to intermediate domain model, and (2) intermediate domain model to database model. Some work specifically focusing on mapping problems from domain model expressions to database target models has been done by [STAL84], [STAL86] and [SCHA82].

The system-specific semantics (i.e., the problem of matching inputs to the appropriate set of relevant hypotheses) has an effect on the structure of the domain model and the problem of lexical translation. The system semantics determine the degree to which synonymous or only nearly synonymous terms should be distinguished. Distinctiveness of terms is determined by the relevance of their semantics to distinguishing a hypothesis' relevance to a given user input.

Synonymous lexical items are of course translated to the same set of predicates. In this system, for example, 'dot' and 'spots' are both translated to a predicate DOTS. The task of hypothesis matching requires additional semantic distinctions to capture the notion of near synonymy. For example, 'streak' and 'slash' are not treated as synonyms at the lowest level in the domain model because for some hypotheses, the distinction between streak and slash is significant. However, user input referring to 'streaks' and 'slashes' may be relevant to hypotheses that describe conditions such as 'dark lines down page.' The near synonymy of 'streak' and 'slash' are defined as subconcepts.

Lexical ambiguity in this system is dealt with by defining more than one lexical translation rule for each lexical item and supplying contexts that must be satisfied before the rule can be applied. In general all input can be sufficiently disambiguated by appealing to the context supplied by the rest of the user input. In occasional cases this is not sufficient. To handle such cases, all predicates have one context-free rule associated with them. This is sometimes necessary for elliptical and ungrammatical input (i.e., cases where the appropriate contest is frequently not present).

The context-free rules guarantee that some translation will always be produced for any user input; however, there are aspects of the translation that are not adequately handled by the current mechanisms used to specify contextually dependent and default rules.

The formal mechanisms of rule contexts and default rules are the primary means of accounting for lexical ambiguity in this system. Other systems propose techniques of constraint satisfaction ([RICH87]) and marker passing ([HIRS84]) to deal with the same types of problems. We have investigated the possibility of assuming a lexical translation rule context (by entering it into a context work area where the partial translation is developed) so that the associated lexical translation rule could apply. If, under the assumed context, a translation for the entire input can be completed, the translation is considered to be correct. If a translation cannot be completed, all translated clauses that depend on the original assumption would be removed and a new assumption made (if necessary). This is similar to the constraint satisfaction approach. We make a distinction not clearly made in either of the cited approaches in the type of contextual information that may be specified in the rules. Lexical translation rules can contain both linguistic contextual information and domain model contextual information. This is useful since some contextual information is more easily specified using one set of terms rather than the other. Currently we have no facility for specifying negative contexts. This facility would occasionally be convenient for the specification of rules.

HYPOTHESIS MATCHING ISSUES

The problem of matching hypotheses to the output of the semantic interpretation component introduces its own set of problems. It is rarely, if ever, the case that the user input will match exactly to any hypothesis representation. It is necessary to specify what kind and how much of a match is needed between the input and the stored hypotheses to justify identifying the hypothesis as a start state for the expert system.

READINESS

EXTENSION TO OTHER SUBDOMAIN PLANS

To fully replace the keyword matching component of the current Page-X interface, we must extend our work to include the other subdomains of the general Page-X problem domain. These include areas of mechanical problems, electrical problems, printing problems, power problems and exceptional cases. This will require additional linguistic extensions and a generalization of the hypothesis identification routine. The overall strategy of the heuristic matching process would remain the same.

LINGUISTIC ROBUSTNESS

Linguistic robustness can be enhanced by conducting experiments to determine sublanguage and by analyzing and making use of the results. Further improvement in the treatment of ungrammatical input is necessary. Currently, there is no technique for handling words that are not in the lexicon. Also, the heuristics employed in assembling a partial interpretation can be made more dependent on linguistic and domain facts. It is possible that the heuristics that are effective for grammatical input will not be as effective for ungrammatical input. This is because an ungrammatical parse will have an effect on the shared variables between predicates in the translated user input. The alfalpha rules and the lexical translation rules can produce new predicates with variables shared across predicates. If default lexical translation rules or nonstandard alfalpha rules must be applied because the parse has not completely succeeded, then there will be cases where two (or more) predicates in the translated user output will not share variables that would have been shared in an equivalent grammatical input translation.

GENERALIZED SYMPTOM DESCRIPTION INTERFACES

It is an open question as to whether or not this interface could be used as the basis for a generalized symptom description interface. The generalization would have to include both the strictly linguistic aspects of the interface and the application-specific aspects. The strictly linguistic portions of the interface (the parser, the three-stage semantic interpretation routines) are applicable to any natural language symptom description interface. The modifications to support partial interpretation of ungrammatical input are also useful in any domain. The application-specific aspects of the interface may be generalizable under restricted conditions. If the new domain is one where 'important concepts' can be identified, then there is a good chance that some version of the matching heuristics could be applied.

BIBLIOGRAPHY

[HIRS84] Hirst, Semantic Interpretation Against Ambiguity, Ph.D. dissertation, Brown University, 1984.

[LARS85] J.A. Larson, W.F. Kaemmerer, K.L. Ryan, J. Slagle, W.T. Wood, "ATOZ - A Prototype Intelligent Interface to Multiple Systems," Foundations for Human Computer Communication, K. Hopper and I.A. Newman (eds), Elsevier Science Publishers B. V. (North Holland) New York, 1986.

[RICH87] Rich, Barnett, Wittenberg, Wroblewski, "Ambiguity Procrastination," Proceedings of AAAI87, July 13-17, 1987, Seattle, Washington, pp 571-576.

[SCHA82] J.H. Remko Scha, "English Words and Databases: How to Bridge the Gap," Proceedings of 20th Annual ACL, June 16-18, 1982, Toronto, Ontario, pp 57-59.

[STAL84] Stallard, "Data Modelling for Natural Language Access," The First Conference on Artificial Intelligence Applications, Denver, Colorado, December 5-7, 1984, pp 19-24.

[STAL86] Stallard, "A Terminological Simplification Transformation for Natural Language Question-Answering Systems," Proceedings of 24th Annual ACL, New York, New York, 1986, pp 241-246.

[STRA85] Strandberg, Abromovich, Mitchell, Prill, "Page-1: A Troubleshooting Aid for Non-Impact Page Printing Systems," Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, Florida, December 11-13, 1985, pp 68-74.

[WITT86] Kent Wittenberg, "A Parser for Portable NL Interfaces Using Graph-Unification-Based Grammars," Proceedings of AAAI86, August 11-15. 1986, Philadelphia, Pennsylvania, pp 1053-1058.