Model-based Analysis of Messages about Equipment

Ralph Grishman, Tomasz Ksiezyk, and Ngo Thanh Nhan

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University

ABSTRACT

The aim of PROTEUS -- a system for the analysis of short technical texts -- is to increase the reliability of the analysis process through the integration of syntactic and semantic constraints, domain knowledge, and knowledge of discourse structure. This system is initially being applied to the analysis of messages describing the failure, diagnosis, and repair of selected pieces of equipment. This has required us to develop a detailed model of the structure and function of the equipment involved. We focus in this paper on the nature of this model and the roles it plays in the syntactic and semantic analysis of the text.

1. Introduction

Considerable progress has been made in developing systems which understand short passages of technical text. Several prototypes have been developed, for such domains as patient medical records [Sager 1978], equipment failure reports [Marsh 1984], and intelligence messages [Montgomery 1983]. Except for very narrow domains such as weather reports, however, none of these systems seem to be robust enough for operational use. Typical success rates - where any are reported - are in the range of 70 to 80% of sentences correctly analyzed; substantially better rates are very hard to obtain, even with careful system tuning[1].

Our objective in developing PROTEUS (the PROtotype TExt Understanding System) is to see if this rate can be substantially improved for a domain of moderate complexity. In order to achieve this improvement, we must bring to bear on the language analysis task the various syntactic, semantic, and discourse constraints, along with a fairly detailed knowledge of the domain of discourse. Our system is initially being applied to equipment failure reports ("CASREPs") for selected equipment on board Navy ships (initially, the equipment in the starting air system); a sample message is shown in Figure 1. In this case, the domain knowledge is the knowledge of the structure and function of these pieces of equipment.

In this paper we first present an overview of the PROTEUS system. We then focus on the domain information: how it is represented, how it is integrated with the language processing, and how it serves to resolve ambiguities in the input text.

2. Prior work

New York University has been involved in the automated analysis and structuring of technical text for over a decade. Most of this work has been on medical records [Sager 1978, Hirschman 1982], but we have also been involved with the Naval Research Laboratory on a system for CASREP's [Marsh 1984]. These systems used domain-specific frame-like target structures, and employed selectional constraints to weed out bad parses, but did not incorporate detailed domain models. Our experience with these systems - in particular, the difficulty of obtaining success rates (% of sentences correctly analyzed) much above 75% - led us to our work on PROTEUS.

The use of detailed domain models in language processing systems is, of course, not new. Script-based systems, and some of the frame-based language analysis systems, have been motivated by a desire to incorporate detailed domain knowledge. The task we confront, however, differs in several regards from those of earlier systems. One is the matter of scale; our initial set of equipment - the starting air system for a gas turbine - includes several hundred separately nameable components (and many lesser components, such as bolts and

---

DURING NORMAL START CYCLE OF 1A GAS TURBINE, APPROX 90 SEC AFTER CLUTCH ENGAGEMENT, LOW LUBE OIL AND FAIL TO ENGAGE ALARM WERE RECEIVED ON THE ACC. (ALL CONDITIONS WERE NORMAL INITIALLY). SAC WAS REMOVED AND METAL CHUNKS FOUND IN OIL PAN. LUBE OIL PUMP WAS REMOVED AND WAS FOUND TO BE SEIZED. DRIVEN GEAR WAS SHEARED ON PUMP SHAFT.

Figure 1. A sample CASREP about a starting air compressor (SAC).

---

[1]Substantially better rates have been cited for strongly expectation-based parsers, which are considered successful if they locate all the expected items within an input text.

gear teeth, without specific names). While not raising any intrinsic difficulties, a domain of this size clearly provides a more rigorous test of our ability to acquire and organize domain knowledge than did many earlier "toy" domains.

Another unusual aspect is the *nature* of the domain information. Scripts, for example, encode essentially procedural information (how to perform complex actions). The information for our domain, in contrast, is primarily structural (part-whole relationships, interconnections, etc.) and to a lesser degree functional. This difference is reflected in differences in the way the information is used - in particular, in the analysis of noun phrases, as we shall see below. Our domain information bears greater resemblance to that used in some equipment simulation packages (e.g., STEAMER [Hollan 1984]) and diagnosis packages [Cantone 1983] than it does to that conventionally seen in natural language systems.

The domain knowledge plays a role in many phases of the language processing task: in the recovery of implicit operands and intersentential relations, in the analysis of noun-phrase reference, and in the determination of syntactic and semantic structure. In particular, we shall consider below its role in the processing of compound nominals, which appear frequently in such technical domains. There have been several prior studies of the processing of such compounds. The work both of Brachman [1978] and of McDonald and Hayes-Roth [McDonald 1978] emphasized the use of search procedures within semantic networks to identify the wide variety of implicit relations possible with compound nominals. We have also used network search techniques, although of a more directed sort. However, their work cited isolated examples from a variety of areas to show the generality of their approach, while we have been concerned with achieving detailed and thorough coverage within a narrower domain. Finin [1980, 1986] has sought to develop, within a sublanguage, general semantic categories for the relations and consituents involved in compounds. Although there are some similarities to our classification efforts, he also has aimed at providing a relatively broad and loose set of constraints. In contrast, the detailed knowledge in our equipment model -- provided for several purposes, of which noun phrase interpretation is only one -- make possible much tighter constraints in our system.

3. System overview

The PROTEUS system has three major components: a syntactic analyzer, a semantic analyzer, and a discourse analyzer. The syntactic analyzer parses the input and regularizes the clausal syntactic structure. The semantic analyzer converts this to a "logical form" specifying states and actions with reference to specific components of the equipment. The discourse component establishes temporal and causal links between these states and actions.

Initial implementations have been completed of the syntactic and semantic components, so that we are able to generate semantic representations of individual sentences. The discourse component is still under development, and so will not be discussed further here.

The syntactic analyzer uses an augmented-context-free grammar and an active chart parser. The grammar is generally based on linguistic string theory and the Linguistic String Project English Grammar [Sager 1981] and includes extensions to handle the various sentence fragment forms found in these messages [Marsh 1983]; it is written in a modified form of the Restriction Language used by the NYU Linguistic String Parser [Sager 1975]. Syntactic regularization maps the various forms of clauses (active, passive, relative, reduced relative, fragmentary) into a canonical form *(verb operand1, operand2...)* The regularization is performed by a set of interpretation rules which are associated with the individual productions and which build the regularized syntactic structure compositionally.[2]

[2] The parser and syntactic regularization procedures were developed by Jean Mark Gawron. The regularization procedures were modeled after those developed for a GPSG parser [Gawron 1982], although the generated structures are quite different.

The semantic analysis component consists of two parts: *clause semantics* and *noun phrase semantics*. The clause semantics maps a clause (a verb plus operands which include syntactic case labels) into a predicate with arguments representing a state or action. Each verb and operand belongs to one or more *semantic classes*. Clause semantics relys on a set of pattern-action rules to perform the translation, with one pattern for each valid combination of verb and operand classes. Noun phrase semantics maps a noun phrase into the identifier of the equipment component specified by that phrase. Noun phrase semantics depends heavily on the equipment model, and so will be discussed further in a later section.

(The division between the two parts of semantic analysis is not quite so neat as the foregoing would suggest. Some noun phrases are nominalizations representing states or actions; these are processed by clause semantics. In many noun phrases, some modifiers identify the object and the remainder describe its state. For example, in "broken hub ring gear", *hub* and *ring* identify the gear, *broken* describes its state. We return to this problem in our description of noun phrase semantics below.)

Our long-term objective is to dynamically schedule among the three analysis components (syntax, semantics, and discourse), as is done in some blackboard models. For program development, however, we have found it better to use a sequential organization (first syntax, then semantics, then discourse). In order to have syntactic choices influenced by semantics and discourse, and semantic choices influenced by discourse, each component may generate multiple analyses, some of which are rejected by later stages. Sometime these multiple analyses are transmitted explicitly, as a list of alternatives. More often, however, they are transmitted using a representation neutral with respect to particular features. The output of syntactic analysis is neutral with respect to quantifier scope. It is also neutral with respect to the distribution of modifiers in conjoined noun phrases (for example, in "filter change and adjustment of pressure regulator," whether *filter* modifies *adjustment* and *of pressure regulator* modifies *change)*. Furthermore, it does not assign structure to prenominal adjectives and nouns (so for example, in the phrase "low lube oil pressure alarm" it does not decide whether *low* modifies *lube, oil, pressure,* or *alarm).*

This system development has been conducted in close cooperation with a group at the System Development Corp., Paoli, PA. Their system, PUNDIT [Palmer 1986], is written in PROLOG but has many points of commonality with PROTEUS in terms of overall structure, grammar, and semantic representation. They are involved in future development of several areas, including semantic representation, time analysis, and anaphora resolution, for both the PUNDIT and PROTEUS systems.

4. The equipment model

The equipment model currently serves three functions within our system:

object identification. The noun phrases in the message are matched against the model (by a procedure outlined in the next section) in order to identify the objects referenced in the message. This is important both for syntactic disambiguation and as a prelude to applying domain-specific inferences.

identification of intersentential relations. The identification of these relations (temporal, causal, and others) is important both for disambiguation (of adjuncts and anaphoric references, in particular) and for establishing the meaning of the message as a whole. Much of the information needed for this process - information on the structure of the equipment and the function of its components - is recorded in the equipment model.

display of equipment structure and status. In order to provide some feedback to indicate whether the text was correctly understood, our system displays a structural diagram of the equipment at several levels of detail. Objects mentioned in the text, and changes in equipment status described in the message, can be shown on the display. The information for generating these displays (positions, shapes, etc.) is stored with the equipment model.

The messages refer to relatively low-level components, such as individual gears within the air compressor. We therefore had to constuct a relatively detailed model of the equipment involved. Our model has been developed through a study of the Navy manuals for this equipment.

The model is basically organized as two hierarchies: a type-instance hierarchy and a part-whole hierarchy. The leaves of the part-whole hierarchy are called *basic parts;* the internal nodes (composite objects) are called *systems.* We record for each system the primary *medium* which it provides, conveys, or transforms; in our starting air system, the three media are compressed air, lubricating oil, and mechanical rotation. We have organized our part-whole hierarchy in part along functional lines (rather than purely on physical proximity), grouping together parts which are connected together and operate on the same medium.

Since some parts are identified by their physical location, we provide a *location* field in both *basic part* and *system* nodes. Both types of nodes also have a *function* field, which indicates the effect of this part on the media or other parts. Nodes of specific types may have additional fields; for example, some mechanical components have a *speed* field.

All of the fields just mentioned record permanent characteristics of the parts. In addition, each node has an *operational-status* field, which holds information about a part which is reported in a message.

The model contains a lot of information about equipment structure which is specific to a particular piece of equipment. Some information, however, is more general: for example, that gears have teeth,or that impellors have blades. It would be most uneconomic to have a separate instance of *tooth* for each gear in the model. Instead we create an instance of the teeth for a specific gear when it is referenced in the text. Such very-low-level objects, which are instantiated dynamically as needed, are called *components.*

The equipment model has been implemented using flavors on a Symbolics LISP Machine. Types of objects are represented by flavors; instances of objects are represented by instances of flavors. The part-whole hierarchy and other fields are stored in instance variables. The structure display is performed by procedures associated with the flavors. The equipment model, and its use in the system, are described in more detail in [Ksiezyk 1986].

5. Noun phrase analysis

The syntactic analysis component analyzes the clause structure and delimits the noun phrases, but does not assign any structure to the pre-nominal modifiers. The noun phrase analyzer within the semantic component therefore has a dual role: to determine the structure of the pre-nominal modifiers and to identify the instance in the equipment model named by the noun phrase (or the set of instances, if this phrase could be applied to any of several parts). (Although there are a limited number of instances, it is not possible to record a single name for each part and then interpret noun phrases by simply looking the name up in a table. A single part can be named in many different ways -- depending in part on prior context -- so a full-fledged interpretation procedure is required.)

The noun phrase is analyzed bottom-up using a set of reduction rules. Each reduction rule combines the head of a phrase with some of its modifiers to form a larger constituent. By reference to the model, each rule also determines the set of instances which can be named by the constituent; if the set is empty, the application of the rule is rejected. Reductions are performed repeatedly until the entire phrase is reduced to a single constituent. If no such reduction is possible, the syntactic analysis is rejected; in this way noun phrase semantics can weed out some incorrect syntactic analyses.

The applicable reductions are determined by the dictionary entries for the words in the noun phrase. Each word is assigned two properties, its *model class* and its *semantic class*. The model class indicates how the word can be related to some entity in the domain model. One value of model class is *instance*, specifying that the word names a set of instances in the model; this set is also included in the dictionary entry. Examples are "pump", "shaft", "gear", etc. Larger constituents built while analyzing the noun phrase are also considered to be of type *instance*. One reduction rule allows us to combine two instances:

$$\text{instance} + \text{instance} \rightarrow \text{instance}$$

for example, "LO" + "PUMP" → "LO PUMP", "SAC" + ("LO PUMP") → "SAC LO PUMP". The set of model instances for the result consists of those instances of the second constituent which can be linked through some path in the model to some instance of the first constituent. The types of links traversed in the search are a function of the *semantic class* of the first constituent; for example, "SAC" has the semantic class *machinery*, so we search the part/whole links, the location links, and the from/to links (which tie together components of the same system).

There are several other model classes and corresponding reduction rules. The class *slot-filler* is used for words which are values of features of instances, but are not themselves instances (for example, "LUBE" in the phrase "LUBE OIL"). The class *slot-name* is used for words which correspond to feature names, such as "SPEED" in "HIGH SPEED ASSEMBLY". The class *component* is used for parts which (as explained in the previous section) are not instantiated in the permanent equipment model but can be instantiated dynamically as needed.

Modifiers describing the state of a part, such as "cracked" or "sheared", are handled differently. If noun phrase semantics gets the input "sheared ring gear" it will look for an instance of ring gear with the *operational-state* "sheared". Such an instance would be present if a *previous* sentence had mentioned that a gear was sheared. If such an instance is found, it is identified as the correct referent; noun phrase semantics has in effect done anaphora resolution. If no instance is found, noun phrase semantics returns the instances of "ring gear" and the left-over modifier "sheared". Clause semantics (which invokes noun phrase semantics) then treats this like a clause "ring gear was sheared"; later in the processing of this sentence, this will cause "sheared" to be assigned as the operational-state of ring gear.

A related technique can be used to handle some of the ambiguities in cojoined noun phrases. For example, in the sentence "INVESTIGATION REVEALED STRIPPED LO PUMP DRIVE AND HUB RING GEAR", syntax alone cannot determine which of the modifiers "STRIPPED", "LO", "PUMP", or "DRIVE" also modify "HUB RING GEAR". So syntax marks these as *possibly* applicable to "HUB RING GEAR" and passes the phrase to semantics. If semantics finds that some of these modifiers cannot be integrated into the noun phrase, they will be ignored, thus implicitly resolving the syntactic ambiguity.

6. Conclusion

We have described a new text-processing system, PROTEUS, for analyzing messages about equipment failure. We have focussed on its equipment model and the role of this model in the process of interpreting of noun phrases. This process is part of semantic analysis but also plays a role in syntactic analysis and discourse analysis.

In addition to the elaboration of the existing components, substantial work will be required in at least two areas before we can hope to obtain a robust text processing system. First, we are developing a discourse component to identify temporal and plausible causal links between sentences. This information is needed not only for some of the applications (e.g., message summarization) but also to resolve some of the syntactic and semantic ambiguities in the messages. Second, we will need to move from a pass/fail strategy for enforcing our constraints to a best-fit strategy. Because of imperfections in the input, and the inevitable omissions in a model as complex as ours, we must expect that many messages will violate one or another constraint; by employing a rich set of constraints, however, and selecting the analysis which violates the fewest constraints, we beleive that we will be able to identify the intended reading for most sentences.

The initial motivation for the system has been the conversion of a stream of messages to a data base for subsequent querying, summarization, and trend analysis. However, the use of a detailed equipment model, similar to that employed in simulation and diagnostic systems, suggests that it may be equally useful as an interface for such systems. A diagnostic system, for example, would then be able to accept initial observations in the form of a brief textual summary rather than force the user to go through an elaborate questionnaire; this may be a substantial advantage for broad-coverage diagnostic systems, which must be able to accept a wide variety of different symptoms.

Acknowledgement

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00014-85-K-0163 from the office of Naval Research, and by the National Science Foundation under grant DCR-85-01843.

REFERENCES

[Brachman 1978] Brachman, R. A. A structured paradigm for representing knowledge. Tech. Rep. No. 3605, Bolt Beranek and Newman Inc., Cambridge, MA.

[Cantone 1983] Cantone, R., Pipitone, F., Lander, W. B., and Marrone, M. Model-based probabilistic reasoning for electronics troubleshooting. *Proc. Eighth Intl. Joint Conf. Artificial Intelligence,* Karlsruhe, West Germany.

[Finin 1980] Finin, T. The semantic interpretation of compound nominals. *Proc. First National Conf. on Artificial Intelligence.* Stanford, CA., Am. Assn. of Artificial Intelligence.

[Finin 1986] Finin, T. Nominal compounds in a limited context. In *Analyzing Language in Restricted Domains,* R. Grishman and R. Kittredge, Eds. Lawrence Erlbaum Assoc., Hillsdale, NJ.

[Gawron 1982] Gawron, J. M., King, J. J., Lamping, J., Loebner, E. E., Paulson, E. A., Pullum, G. K., Sag, I. A., and Wasow, T. A. Processing English with a generalized phrase structure grammar. *Proc. 20th Annual Meeting Assn. Computational Linguistics,* Toronto, Canada.

[Hirschman 1982] Hirschman, L., and Sager, N. Automatic information formatting of a medical sublanguage. In *Sublanguage: Studies of Language in Restricted Domains,* R. Kittredge and J. Lehrberger, Eds. Walter de Gruyter, Berlin.

[Hollan 1984] Hollan, J., Hutchins, E., and Weitzman, L. STEAMER: an interactive inspectable simulation-based training system. *AI Magazine,* Summer 1984, 15-27.

[Ksiezyk 1986] Ksiezyk, T. An equipment model and its role in noun phrase interpretation. Submitted to *ACM Conf. on Object Oriented Programming Systems, Languages, and Applications,* Portland, OR.

[Marsh 1983] Marsh, E. Utilizing domain-specific information for processing compact text. *Proc. Conf. Applied Natural Language Processing,* Santa Monica, CA.

[Marsh 1984] Marsh, E., Hamburger, H., and Grishman, R. A production rule system for message summarization. *Proc. 1984 National Conf. on Artificial Intelligence,* Austin, TX, Am. Assn. of Artificial Intellgience.

[McDonald 1978] McDonald, D., and Hayes-Roth, F. Inferential searches of knowledge networks as an approach to extensible language understanding systems. In *Pattern-directed inference systems,* Waterman and Hayes-Roth, Eds. Academic Press, New York.

[Montgomery 1983] Montgomery, C. Distinguishing fact from opinion and events from meta-events. *Proc. Conf. Applied Natural Language Processing,* Santa Monica, CA.

[Palmer 1986] Palmer, M., Dahl, D., Schiffman, R., Hirschman, L., Linebarger, M., and Dowding, J. Recovering implicit information. To appear in *Proc. 1986 Annl. Conf. Assn. Computational Linguistics,* New York, NY.

[Sager 1975] Sager, N., and Grishman, R. The restriction language for computer grammars of natural language. *Comm. Assn. Computing Machinery 18,* 390-400.

[Sager 1978] Sager, N. Natural language information formatting: the automatic conversion of texts to a structured data base. *Advances in Computers 17,* 89-162.

[Sager 1981] Sager, N. *Natural Language Information Processing.* Addison-Wesley, Reading, MA.