

AN APPLICATION OF AUTOMATED LANGUAGE UNDERSTANDING TECHNIQUES TO THE GENERATION OF DATA BASE ELEMENTS

Georgette Silva, Christine Montoomerv. and Don Dwigings
Operating Systems, Inc.
21031 Ventura Boulevard
Woodland Hills, CA 91364

This paper defines a methodology for automatically analyzing textual reports of events and synthesizing event data elements from the reports for automated input to a data base. The long-term goal of the work described is to develop a support technology for specific analytical functions related to the evaluation of daily message traffic in a military environment. The approach taken leans heavily on theoretical advances in several disciplines, including linguistics, computational linguistics, artificial intelligence, and cognitive psychology. The aim is to model the cognitive activities of the human analyst as he reads and understands message text, distilling its contents into information items of interest to him, and building a conceptual model of the information conveyed by the message. This methodology, although developed on the basis of a restricted subject domain, is presumed to be general, and extensible to other domains.

Our approach is centered around the notion of "event", and utilizes two major knowledge sources: (1) a model of the sublanguage for event reporting which characterizes the message traffic, and (2), a model of the analyst-user's conceptualization of the world (i.e., a model of the entities and relations characteristic of his world).

THE SUBLANGUAGE

The two sublanguage domains studied thus far consist of descriptions of events involving aircraft activities and launchings of missiles and satellites.

The source data are contained in the text portions of military messages typical of these subject domains, consisting of a report title summarizing a given event, followed by one or more declarative sentences describing that event (and optionally, other related events).

Both the semantics and the syntax of these event descriptions are constrained by two factors. One, by the particular subject domain, and two, by the fact that the events described are limited to what is observable and what should be reported according to a reporting procedure. This results in a substantial number of participial constructions of various types, complex nominalizations and agentless passives, as well as a range of types of quantification, conjunction, complementation, ellipsis, and anaphora. The sublanguage, although less extensive in its inventory of syntactic constructions than event reports in journalistic narrative, nevertheless contains certain constructions which present challenging semantic problems. Such problems include the treatment of "respectively" constructions, as well as certain types of definite anaphora which not only transcend sentence boundaries and, in some cases, even message boundaries, but often are of the kind that have no explicit referent in the previous discourse.

Of the two languages studied thus far, the discourse structure of the missile and satellite reports is considerably more complex than that of air activities. While in air activities reports the description of a given event is often completed within a single sentence

(e.g., a particular aircraft penetrated enemy airspace at a specific location and a specific time), in missile and satellite reports the complete specification of the properties of an event and of the object(s) involved more frequently requires several sentences, and not uncommonly, several messages. Thus, a report on some launch operation can consist of an initial, rather skeletal statement, followed by one or more messages received over a period of time, which update the previous report, adding to and sometimes changing previous specifications. The boundaries of a discourse relevant to a single event, then, can range from a single sentence to several messages. The problem of assembling the total mental "picture" relating to any given event can only be approached on the discourse level.

Any message may contain descriptions of more than one event. These events may be connected in some way, or totally unrelated (e.g., a summary). Our approach to this problem is to describe the meaning content of the message in terms of a "Message Grammar" in which the "primitives" are event classes, and the relations are discourse-level relations. The latter may be optional or obligatory and determine the connectivity or non-connectivity between events.

THE WORLD MODEL

A particular world of discourse is characterized by a collection of entities, including their properties and the relations in which they participate. We define a world model in terms of abstract data structures called "templates", which resemble linguistic case frames. Each template describes a class of entities in terms of those properties which are normally associated with that class in a particular domain. A template thus reflects the information user's conceptualization of the domain, i.e., his view of what that class of entities involves. In the domains under investigation there are templates for classes of objects (aircraft, missiles), classes of events (flights, launchings), classes of relations (temporal, causal), and other concepts such as time and date. A template represents an n-ary relation, where the n-ary relationship is named by a predicate symbol (e.g., Precede (Event₁, Event₂), Enroute (Object, Source, Destination, Time, etc.)).

The templates are the basic data objects of an Event Representation Language (ERL), an experimental language written to explore the use of "templates" as a knowledge representation technique with which to build language understanding systems for message text analysis.

The Event Representation Language is implemented in a subset of Prolog, a formalism using a clausal form of logic restricted to "Horn" clauses. Horn clauses can be given both a declarative and a procedural interpretation and are therefore very well suited for the expression of concepts in the Event Representation Language. The basic computational mechanism of Prolog is a pattern matching process ("unification") operating on general record structures ("terms" of logic).

Templates are encoded as "construct" clauses. For example, the DEPLOY template, which is informally

This research was sponsored by the Air Force Systems Command's Rome Air Development Center, Griffiss Air Force Base, New York.

Table 1. Informal Description of the DEPLOY Concept

Descriptive Elements			Procedural Elements
Descriptor	Filler Specification	OBL / OPT	Procedures for filling slots
Object	Logical Subject noun phrase (+acft)	OBL	Construct 'Aircraft' template from logical subject
Destination	PP: 'to'+ NP(+loc)	OBL	Search VMODS list for appropriate prepositional phrase
Time	1. Adv(+ time + ref) 2. PP (during, between +NP(+ time))	OPT	Search VMODS list for appropriate constituent

Table 2. Prolog Representation of DEPLOY Template

```

construct ('DEPLOY', s(Subj,Vbgr,Obj,Comp1,Vmods),[OB1,S1,L2,DTG]):-
    object(Subj,OB1),
    destination(Vmods,D1),
    construct('DTG',Vmods,DTG).
    
```

Table 3. A "Destination" Clause

```

destination(Vmods, slot('DESTINATION=',Slot)):-
    fill-slot(Vmods,['TO'],'LOC',Slot).
    
```

represented in Table 1 in a simplified form, is encoded as in Table 2.

The head of the "construct" clause has three arguments: a template name, the name of the syntactic constituent which serves as the context which is searched in an attempt to find fillers for the descriptor slots of the template in question, and a third argument which represents the output of the procedure, i.e., the instantiated slots.

The body of the "construct" clause consists of three "goals" corresponding to the three slots of the DEPLOY template shown in Table 2. These three goals are themselves defined as procedures, which seek fillers for the descriptor slots they represent.

For example, the "destination" slot in the "construct" procedure for DEPLOY is written as in Table 3.

This representation has certain advantages, among which we might mention the following two: (1) if additional information needs to be associated with a particular predicate, this can be done simply by adding another clause; and (2), Prolog provides a uniform way of representing structures and processes at several levels of grammatical description: syntactic structures, syntactic normalization, description of objects, description of events, and description of text-level relations.

THE UNDERSTANDING PROCESS

The formal definition of the sublanguage currently takes the form of an ATN grammar. The parser takes a sentence as input and produces a parse tree. The parse is input to the ERL "machine", which uses templates for the interpretation of the input and produces "event

records" as output. Event records can be viewed as "instantiated" templates. They are event-centered data structures in which the information conveyed by the input can be viewed from the perspective of time, location, type of activity, object(s) involved, etc. These event records constitute the "extensional" data base which serves as a support tool for higher-level analytical functions in a decision-making environment.

The computer program which embodies this approach to natural language understanding is written in FORTH, Prolog, and SNOBOL4, and runs on a PDP 11/45 under the RSX operating system.

The major part of the system was built in the programming language FORTH, which is an interactive, incremental system with a low-level semantics which the user can easily and quickly extend. This allowed the rapid development of the ATN language and control scheme, as well as the support scheme for the execution of the ERL algorithms. These are written in Prolog, which is--as mentioned above--a language that is well suited to the specification of templates and the algorithms for instantiating them. For ease of implementation, the compiler for the subset of Prolog utilized in this application was written in SNOBOL4.

The use of FORTH and the Prolog formalism allowed fairly easy development of the system even without the powerful structure manipulation capabilities of a language like LISP. The major impact of the minicomputer environment was felt near the completion of system development, when the combined programs nearly filled the available 64K byte address space. This has been mitigated somewhat by moving the working data to a form of virtual memory which is supported by FORTH, and by overlaying the grammar code with the interpretation code.

