COGNITIVE SCIENCE 3, 251–273 (1979)

Prediction and Substantiation: A New Approach to Natural Language Processing*

GERALD DEJONG

Yale University

This paper describes a new approach to natural language processing which results in a very robust and efficient system. The approach taken is to integrate the parser with the rest of the system. This enables the parser to benefit from predictions that the rest of the system makes in the course of its processing. These predictions can be invaluable as guides to the parser in such difficult problem areas as resolving referents and selecting meanings of ambiguous words. A program, called FRUMP for Fast Reading Understanding and Memory Program, employs this approach to parsing. FRUMP skims articles rather than reading them for detail. The program works on the relatively unconstrained domain of news articles. It routinely understands stories it has never before seen. The program's success is largely due to its radically different approach to parsing.

1.0 INTRODUCTION

People are often able to make very reliable predictions while understanding a sentence. For example, consider the following sentence fragment:

(1) Yesterday there was an earthquake in . .

One expects the next word or phrase to be a location like "California" or "Venezuela," before ever seeing the word.

The utility of such predictions is obvious. If characteristics of words can be predicted before the words themselves are seen then these predictions can be used to choose among word senses and guide any necessary inferencing. What rules govern these predictions? When and how are they made? In particular, where does this prediction come from?

*This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under contract N00014-74-C-1111.

One possibility is that the prediction is stored in the dictionary definition of "in." The definition might include that the object of the preposition is usually a location. But this is clearly wrong. The preposition "in" is frequently used to specify a time duration of an event (e.g., "The game was over in half an hour"). Thus there ought to be equally strong predictions for a location and a time duration to follow (1). This is not the case. In the sentence

(2) Yesterday there was an earthquake in three minutes,

the phrase "in three minutes" is difficult to understand. Surely a time duration is not predicted. One might be able to interpret (2) as "Yesterday there was an earthquake somewhere which lasted three minutes" but even this reading is forced.

Another possibility is that the word "earthquake" predicts that it will be immediately followed in the sentence with a location. The word "in" is then interpreted as just the first word of the expected locative phrase. This accounts for why (2) is bad, but this rule also breaks down. Consider

(3) An earthquake struck Mexico.

Here "earthquake" is immediately followed by the verb "struck."

Clearly the correct reason for the prediction somehow depends on both "earthquake" and "in." Neither word alone is sufficient to reliably predict a location.

A rule like "if the word 'earthquake' is followed by the word 'in' then expect a location" is also wrong. This rule amounts to adding a phrasal dictionary definition for "earthquake in" which expects a location to follow it. While it seems to work, the rule is too specific to these words. The phenomenon appears to be much more general than is captured by this rule. After all the same prediction ought to be made from "earthquake near," "tremor in," and a dozen other phrases.

The above prediction methods fail because they are too word oriented. The location prediction is derived from the background context of earthquakes, not just the words "earthquake" and "in." This information is not likely to be contained in any dictionary definition. Rather it is static knowledge about earthquakes in general, irrespective of the words used to describe the situation (earthquake, tremor, etc).

Consider instead the following more general approach to making such predictions. The situational context brought up, in this case by the word "earthquake," expects several additional data one of which is a location. Since the word "in" has a reading which can identify a location, that reading is preferred. It is the interaction between the situational knowledge about earthquakes (that they occur at locations) and the linguistic knowledge about the word "in" (that it can add a location) which yields the prediction.

To design a natural language system that takes advantage of such predictions, we need a system organization that allows world knowledge to directly

affect the process of text analysis (assigning meanings to words). Previous systems did not allow this kind of interaction. In the next section we will first explore a more forceful argument for this kind of interaction. Then we will discuss why such interaction is not possible with the approach taken by previous natural language systems.

2.0 DEFICIENCIES OF PREVIOUS SYSTEMS

To illustrate how crucial context can be in text analysis, consider the sentence "He took it." This sentence can have many different meanings in different contexts:

(4) Mary told John it was time for his medicine. He took it.

(5) The batter prepared for the pitch. He took it (low and outside).

(6) John saw that Bill's bishop was en prise. He took it.

(7) Bill gave John some very sound advice. He took it.

In each of the above sentences, the sentence "He took it" has a different meaning depending on the context provided by the previous sentence. No text analyzer can produce the correct meaning representation of the second sentence without the contextual knowledge provided by the first.

In systems with syntactically oriented front ends, the problem does not arise in this example. Syntactically, the sentence "He took it" is unambiguous. It is the meaning that is unclear. These systems produce only a syntactic parse and let some later semantic process assign the meaning. However, these systems suffer when semantics is necessary to prefer a syntactic parse as is often the case for prepositional phrase attachment. The sentence

(8) Bill hit the boy with a broken leg.

is ambiguous on both the syntactic and semantic levels: the boy could have a broken leg or Bill could be using a broken leg as a club. The former reading is clearly preferred to the latter on semantic grounds. However, one can imagine a bizarre context in which a wooden table leg might be used as a club. A purely syntactic parser with no access to contextual information cannot eliminate the need to produce both parses.

Thus whether the text analyzer must do semantic analysis or is just producing a syntactic parse, it must have access to contextual information.

Most previous natural language understanding programs have been made up of at least two separate subsystems (Cullingford, 1978; Kaplan, 1975; Schank, 1975; Wilks, 1973; Woods, 1970). A *parser* subsystem analyzes the input natural language text into some intermediate form. The intermediate representations range from surface representations like syntactic parse trees (Thorne et al., 1968; Marcus, 1977) and simplified English (Parkinson et al., 1976) to

representations involving conceptual primitives (Riesbeck, 1975). An *inferencer* subsystem then builds a representation of the meaning of the input text. This involves incorporating the parser output into the meaning representation, inferring any missing events, and supplying the causal connections between events. Most natural language programs include various other subsystems as well—such as a question answerer or a summarizer. These are used to demonstrate understanding and do not affect the understanding process.

Figure 1 Block diagram of a conventional natural language system.

Whatever the form of the intermediate representation, its purpose is always the same: to insulate the inferencer from the capriciousness of natural language. In all of these systems, however, one gets the feeling that there are actually two parsers: the one everyone admits to having which produces the intermediate representation, and the second one hidden in the "inferencer" which parses the output of the first parser to decide what it really means.

Natural language systems organized as in figure 1 cannot easily make predictions like the earthquake-location prediction discussed above because the necessary context is not available to the parser. In these systems, the parser processes one sentence at a time. The intermediate representation built from the sentence is given to the inferencer. The inferencer then incorporates the new representation into the internal structure that it built from previous sentences. Thus the processing is all one way.

Not all previous systems disallow communication from the inferencer to the parser. Systems proposed by Marcus and Winograd permit communication both ways between the modules.

In SHRDLU (Winograd, 1972) a systemic grammar is used to parse the input. The systemic grammar is a type of syntactic grammar. However, Winograd permits the grammar rules to call semantic interpretation functions. When a syntactic entity (e.g., a noun phrase) is parsed it can be immediately analyzed by the semantic interpreter. If the semantic interpreter cannot make sense of it, the phrase can be reparsed. Information never flows from semantics to syntax except for a "go/no-go" evaluation of what syntax produced.

Marcus (1977) takes a very similar but more general view in his parser. He does envision communication between his syntactic analyzer and proposed semantic and pragmatic modules. However, he emphasizes that the semantic and pragmatic modules are called only at the request of the syntactic analyzer. Thus

he allows more than just "go/no-go" information, but again the information flow can only be initiated by the syntactic module.

Even though systems like Winograd's and Marcus's permit more communication between the parser and inferencer during processing, it is the wrong kind of communication. Semantic analysis is still done only after syntactic analysis. Semantic context is not used to provide the parser with information that can help it along. Rather semantic analysis is done only at the request of the syntactic module after a certain amount of processing. Both systems defer semantic analysis until syntactic processing indicates that it is appropriate. In Winograd's SHRDLU this is done when a syntactic entity has been found; in Marcus's when his syntax analyzer reaches a choice point.

The SOPHIE system (Brown & Burton, 1975) allows the right kind of context/parsing interactions. This results in, not surprisingly, a very robust system. However, the system organization fundamentally restricts the program to exactly one every constrained domain.

SOPHIE uses a "semantic grammar" (Burton, 1976) to analyze natural language input. The rules in the semantic grammar try to recognize entities with certain semantic properties rather than syntactic ones. This is done by incorporating much of the world knowledge of the domain into the grammar rules. A very robust and successful system results which can handle inputs with deletions, ellipses, and anaphoric referents. The price paid by the semantic grammar approach is to make the language analyzer extremely domain dependent. The parser is closely tied to a very constrained microworld. Extensive rewriting of the grammar would be necessary to change domains. While the resulting system is very impressive, the paradigm is so confining in terms of its natural language capabilities that it is questionable what the system has to say about natural language processing in general.

In summary, context must be supplied to the text analyzer for it to do its job efficiently. With the exception of the SOPHIE system, previous natural language systems denied context to their text analyzers. This is the root cause of previous systems' inability to process novel unanticipated inputs. The SOPHIE system, on the other hand, solved the problem at the expense of domain flexibility. It is an efficient and robust system but it is constrained to natural language inputs concerned with debugging the Heathkit IP-28 power supply.

3.0 AN IMPROVED SYSTEM ORGANIZATION

I will now propose a system organization which has the best of both worlds. It permits contextual knowledge to influence parsing (as does SOPHIE) and keeps domain-specific world knowledge separate from linguistic knowledge (as the others do). The proposed system organization also has two modules: a PREDIC-

TOR and a SUBSTANTIATOR. One module predicts constraints on what might happen next. The other module tries to justify and give substance to these predicted characterizations of the next possible events. The communication between the two modules is comparatively unconstrained.

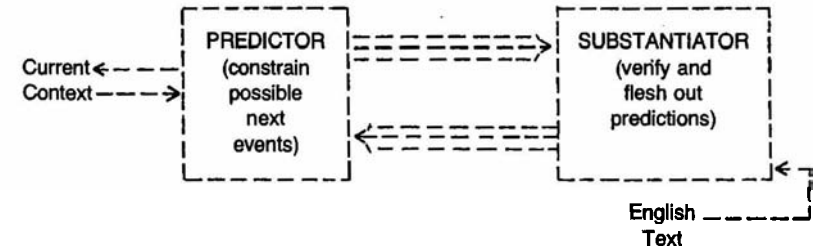

Figure 2 An improved system organization.

This decomposition is radically different from figure 1. One might roughly identify the SUBSTANTIATOR with the PARSER and the PREDICTOR with the INFERENCER. But as we shall see, the above system organization is fundamentally different.

The PREDICTOR jumps to conclusions about what the text means. It predicts characteristics of possible next events in the text.

These predictions are based on the *current context*. The current context is the representation of the previously processed text from the article. When a prediction is substantiated, it is added to the current context. Thus substantiated predictions can be used for the basis of further predictions.

All of the PREDICTOR's characterizations of possible next events are on the conceptual level. The PREDICTOR cannot anticipate actual input words. Rather, it predicts conceptual representations. A conceptual representation might correspond to many differently worded phrases.

The predictions can be at any conceptual level; the module can predict an entire conceptualization or a small piece of a conceptualization. Furthermore, multiple parallel predictions can be made. Thus, the PREDICTOR might hypothesize that the actor of the next input event will be either a head of state or a military spokesman, or it might make a guess at an entire conceptualization such as—the next event will describe Israeli bombers attacking Palestinian positions.

Because of the structure of English texts the PREDICTOR cannot anticipate all and only those conceptualizations which will appear in the input. Natural language texts often leave out important conceptualizations. The PREDICTOR hypothesizes what the next events might be—based on what has been understood so far. Thus the PREDICTOR anticipates probable conceptual entities. Some of the conceptualizations will be explicitly present in the text while others must be inferred.

Of course, the predictions can be wrong. When they are the PREDICTOR must reassess the assumptions on which the predictions were based. This reassessment can result in different predictions. If, on the other hand, a particular prediction proves correct, it is added to the current context. In this way the most up-to-date information is available for further predictions. The final internal representation of an article is the current context at the time the last word of the article is processed.

The SUBSTANTIATOR tries to verify predictions made by the PREDICTOR. Verification can be done either by finding a text input that matches a prediction, or by making an inference (based on what has already been understood) that matches a prediction.

The SUBSTANTIATOR must usually flesh out the predictions into conceptual representations of actual events since the raw predictions are often the only general characteristics of events.

The process of giving substance to the predicted characterizations is very purposeful. Text is analyzed and inference routines are called only in response to the PREDICTOR's anticipations of events. Thus the SUBSTANTIATOR's processing is channeled by the PREDICTOR's processing.

In this scheme, text analysis is driven by the PREDICTOR, not by the input. There is no conventional parser which produces conceptualizations when it is presented with a sentence. Rather, the input text is analyzed only when the PREDICTOR wants a specific piece of information and the SUBSTANTIATOR has decided that the missing information might be found in the text. The text is only examined a little bit at a time, and then only to substantiate specific predictions.

In the PREDICTOR/SUBSTANTIATOR organization, there is a problem in getting the system to start reading. When the system is presented with a new input text there is no current context, so no predictions can be made. Obviously, there must be some way to initially build up the current context to enable predictions. That is, the system must do some bottom-up processing before it can utilize its top-down knowledge.

This problem is not unique to the PREDICTOR/SUBSTANTIATOR method of processing natural language. It is common to all predictive understanders. Every script/frame/schema system has the problem of identifying the correct script/frame/schema to use in a given situation.

There is no general solution to this problem. Each predictive understander must find a solution based on its own needs. Bobrow et al. (1977), Charniak (1977), and Cullingford (1978) each proposed a specific solution. However, these solutions are either too narrow or inefficient for a PREDICTOR/SUBSTANTIATOR natural language system. See DeJong (1979) for a detailed discussion of why their solutions are not acceptable and a description of a more general solution for a particular PREDICTOR/SUBSTANTIATOR system.

In this section, we have seen that communication between the PREDICTOR and SUBSTANTIATOR modules assures that they are both operating in the richest

and most constrained context possible. When the SUBSTANTIATOR verifies a prediction, it tells the PREDICTOR the actual conceptualization it found. On the basis of the additional information included in the fleshed out conceptualization, the PREDICTOR can refine its predictions. It might be able to make a previous prediction more explicit, it might take back previous predictions, and it might make new ones. These refined predictions are communicated to the SUBSTANTIATOR to redirect its efforts. Thus the text analyzer and the inference processor of the SUBSTANTIATOR are always operating in the most complete context that the PREDICTOR can provide.

4.0 WHY IS THIS DECOMPOSITION MORE NATURAL?

Recently inferencers have become more and more predictive in nature. Compare, for example, the bottom-up MARGIE inferencer (Rieger, 1975) to the SAM script applier (Cullingford, 1978) or consider the top-down approach taken by the HEARSAY I system (Reddy et al., 1973). It has become increasingly clear that an inferencer must know what kinds of inputs to expect in order to make sense of them. The existence of so many "framelike" systems illustrates this (Bobrow et al., 1977; Charniak, 1977; Cullingford, 1978; Goldstein & Roberts, 1977; Wilensky, 1978). While a frame is a very broad concept (Minsky, 1975), all framelike systems have one thing in common: they are largely top-down processors. That is, to facilitate the understanding process they predict what inputs will look like before they are seen.

And yet, in natural language processing there has been little attempt to take advantage of the inferencer's predictions when parsing natural language text. The conventional design of natural language systems makes effective communication between the inferencer and the parser extremely difficult.

5.0 BENEFITS OF THE NEW ORGANIZATION

Now that we have briefly explored the new organization and what motivated it, we will discuss advantages this system has over previous ones. The advantages are primarily increased efficiency and robustness. Viewing natural language processing in this way also helps to solve the problem of anaphora.

5.1 Increased Efficiency

Predictions on the conceptual level help system efficiency in resolving ambiguous words. Word senses are chosen which help to substantiate a prediction. Consider the following story:

> There was renewed fighting today between Israeli and Syrian forces. Syrian soldiers fired mortars at Israeli positions in the Golan Heights.

Suppose our understander is reading this story about a military engagement between two countries. We will concentrate on how the system processes the word "fired" in the second sentence. Processing the first sentence results in a current context representing a fighting situation between two countries, Israel and Syria.

In reading the second sentence, the word "fired" is encountered. The word "fire," even when known to be a transitive verb, has several meanings. It might mean "shoot" but it might also mean "put pottery into a kiln" or "terminate employment." Furthermore, the meaning cannot be decided from syntactic cues alone. One might guess that "fired" when followed by the preposition "at" always means "shoot." But this is not the case: "Syrians fired mortars at Israeli positions," "Bill fired clay pots at 300 degrees," and "The boss fired John at three this afternoon" all have the preposition "at" following different word senses of "fired."

Of course, in the context of a story about a battle, the word "fire" ought to be interpreted as "shoot." In our new system organization using the current context built by the first sentence, the PREDICTOR anticipates that among possible next events are shooting events. When the SUBSTANTIATOR encounters the word "fired," it immediately resolves it to the "shoot" meaning because that is the meaning which satisfies a prediction.

In this scheme, most words are disambiguated as they are encountered. This means that ambiguous parses are not carried along in parallel. This scheme eliminates the wasteful process of trying out different word senses to see if a valid parse will result and leads to much greater efficiency.

5.2 System Robustness

Robustness is achieved by considering syntactic rules as a set of heuristics rather than a rigid grammar. In a system such as this, understanding is primarily driven by semantic considerations. Syntactic information is used by the text analyzer to provide clues about where in a sentence a desired conceptual entity might be found. That is, it mediates between conceptual predictions and sentence locations. In this way syntax can limit searching through the text.

The text analyzer does not need a very complete knowledge of syntax. Text analysis is motivated by predictions of conceptual items. Syntactic knowledge is used only to find the general sentence location of the desired word. The semantic predictions determine exactly which word and word sense will be used to build the predicted conceptual item.

The alternative to syntax heuristics is to provide the system with an explicit grammar. Many other natural language systems have done precisely this (Marcus, 1977; Winograd, 1972; Woods & Kaplan, 1971). There are two disadvantages to having such a grammar. First, the grammar has to be very complicated if it is to account for a large part of English. This makes processing much less efficient. Second, English has eluded every attempt at constructing a rigorous

grammar for it. No matter how complicated the grammar, there are always large classes of English sentences that cannot be parsed. Thus an explicit grammar is overly constraining. By eliminating the need for such a grammar, a main stumbling block to system flexibility and robustness is removed.

5.3 Anaphoric References

The top-down orientation of the PREDICTOR-SUBSTANTIATOR system allows the resolution of a large class of pronominal references easily. Recall from the discussion of the PREDICTOR that its predictions are constantly revised to be the tightest, most accurate possible. At times the PREDICTOR can anticipate the precise filler of a desired role. For example, instead of predicting that the actor of a conceptualization will be filled with a country, it might predict that the actor will be "France." This is not to say that the PREDICTOR has anticipated which word will appear in the text. The prediction is of the internal concept for France. The concept "France" might be specified in the text in other ways than just its proper name. For example a pronoun might be used.

When such a pronoun is encountered, it is resolved to the predicted conceptual item. Thus, these kinds of pronouns are not a problem. In fact, they are an advantage. Pronouns improve processing efficiency because they eliminate the need for a careful matching that would otherwise be performed.

In many systems pronouns are handled differently from other processing. When a pronoun is encountered, a list of possible referents is created. One of these is picked on the basis of gender and number matching and on how well the referent fits semantically into the rest of the input. Often a special routine is called to find a referent for a pronoun although Charniak (1972) argued convincingly against this technique.

To illustrate how a PREDICTOR/SUBSTANTIATOR can process pronouns, consider the pronoun occurring in the second sentence of the following input:

> An Exxon oil refinery was nationalized by Uganda. It was paid $1.2 million in compensation.

After processing the first sentence, the current context of the system will represent one country forcibly taking over a company. The first sentence builds a meaning representation in which there is a forced takeover of an oil refinery. The actor of the takeover conceptualization is Uganda and the refinery used to belong to Exxon. Of course, the system must have a great deal of world knowledge about nationalizations. On the basis of this world knowledge and the context from the first sentence, the PREDICTOR can anticipate that there might be a payment to Exxon from Uganda. While processing the second sentence, the SUBSTANTIATOR finds the word "paid" which builds a structure that partially matches the above prediction. In following up on this partial match, the PREDIC-

TOR predicts that a payment will be made to Exxon. In the sentence location where the SUBSTANTIATOR expects to find the recipient concept, the word "it" is found. Since "it" does not have any properties that contradict the concept Exxon, Exxon is assumed to be the referent for "it."

In this scheme, less processing is required for a pronoun than a normal word. A pronoun eliminates the need for a detailed match to be performed. It says basically "never mind the further processing you would do if there were a real word here, the prediction you have is the correct one provided it matches in gender and number." If in the above example, "the Exxon Corporation" were found instead of "it," the text analyzer would have to justify that "the Exxon Corporation" could indeed be considered to refer to the internal concept for Exxon. Although in this case the justification would be quite easy, it would involve some work, and in general could involve a careful detailed match. Thus, in a PREDICTOR/SUBSTANTIATOR organization, pronouns make text interpretation more efficient rather than problematic.

6.0 THE FRUMP PROGRAM

FRUMP (Fast Reading Understanding and Memory Program) is a computer system that has been built based on the idea that world knowledge should be allowed to affect the interpretation of words. FRUMP skims input text for important facts rather than reading it for detail. The flexibility and robustness of the approach is demonstrated by FRUMP's ability to correctly process English text which has never before been seen by either the program or its programmers. Furthermore, the domain of the program is not excessively constrained. FRUMP is designed to work on news articles. The program can process text from diverse domains such as reports of plane crashes, countries establishing diplomatic ties, forest fires, and wars. New domains are added by supplying only the domain specific world knowledge; FRUMP's linguistic knowledge is independent of any particular domain.

A UPI news wire is connected to the Yale computer to provide real world data for FRUMP. FRUMP routinely understands actual news articles from the UPI news wire and notifies logged-in users by sending a summary to their terminals. The PREDICTOR/SUBSTANTIATOR approach also enables FRUMP to be very efficient. Most news articles are processed in less than 20 seconds of CPU time on a DEC System 20 computer. FRUMP can easily keep up with the rate news stories arrive over the UPI news wire.

FRUMP uses a data structure called a *sketchy script* to organize its knowledge about the world. Each sketchy script is the repository for the knowledge

FRUMP has about what can occur in a given situation. FRUMP currently has sketchy scripts for 48 different situations ranging from earthquakes to countries establishing diplomatic ties to labor strikes.

Scripts have been used before for natural language processing (Schank & Abelson, 1977; Cullingford, 1978). However, they were detailed scripts. A detailed script contains all of the events that might occur in a situation; a sketchy script contains only the important events.

We estimate that about 30 percent of the texts from the news wire are understandable by a script processor such as FRUMP. The remainder consists largely of human interest stories, corrections of previous articles, and information to UPI subscribers for which no script can be written. FRUMP understands about one third of the scriptal stories. That is, about 10 percent of the wire stories. There are several reasons why FRUMP does not attain the theoretical limit of 30 percent. In order of importance they are—lack of world knowledge, lack of vocabulary, and insufficient knowledge about English sentence structure. This is not to say that FRUMP will never achieve the 30 percent figure. There is every reason to expect that once it is out of its research phase FRUMP can be provided with a much larger vocabulary and many more scripts.

7.0 SOME EXAMPLES

The following examples show some of FRUMP's capabilities. FRUMP is a fully implemented system. The same version of the system processed all of the examples shown here. These example stories are all relatively short. Short articles are shown here to conserve space. Longer stories result in very similar summaries. The length of FRUMP's summaries depends more on how detailed FRUMP's world knowledge is for the situation being reported than on the length of the article.

The following story was taken from the New York Times. It demonstrates FRUMP's ability to understand the main thrust of a news story while ignoring the less important details.

INPUT:

WASHINGTON, MARCH 15—THE STATE DEPARTMENT ANNOUNCED TODAY THE SUSPENSION OF DIPLOMATIC RELATIONS WITH EQUATORIAL GUINEA. THE ANNOUNCEMENT CAME FIVE DAYS AFTER THE DEPARTMENT RECEIVED A MESSAGE FROM THE FOREIGN MINISTER OF THE WEST AFRICAN COUNTRY SAYING THAT HIS GOVERNMENT HAD DECLARED TWO UNITED STATES DIPLOMATS PERSONA NON GRATA.

THE TWO ARE AMBASSADOR HERBERT J. SPIRO AND CONSUL WILLIAM C. MITHOEFER JR., BOTH STATIONED IN NEIGHBORING CAMEROON BUT ALSO ACCREDITED TO EQUATORIAL GUINEA.

ROBERT L. FUNSETH, STATE DEPARTMENT SPOKESMAN, SAID MR. SPIRO AND MR. MITHOEFER SPENT FIVE DAYS IN EQUATORIAL GUINEA EARLIER THIS MONTH AND WERE GIVEN "A WARM RECEPTION."

BUT AT THE CONCLUSION OF THEIR VISIT, MR. FUNSETH SAID, EQUATORIAL GUINEA'S ACTING CHIEF OF PROTOCOL HANDED THEM A FIVE-PAGE LETTER THAT CAST "UNWARRANTED AND INSULTING SLURS" ON BOTH DIPLOMATS.

SELECTED SKETCHY SCRIPT $BREAK-RELATIONS

CPU TIME FOR UNDERSTANDING = 2515 MILLISECONDS

ENGLISH SUMMARY:

THE US STATE DEPARTMENT AND GUINEA HAVE BROKEN DIPLOMATIC RELATIONS.

FRENCH SUMMARY:

LE DEPARTEMENT D'ETAT DES ETATS-UNIS ET LA GUINEE ONT COUPE

LEURS RELATIONS DIPLOMATIQUES.

CHINESE SUMMARY:

MEEIGWO GWOWUHYUANN GEN JIINAHYAH DUANNJYUELE WAYJIAU GUANSHIH.

SPANISH SUMMARY:

EL DEPARTAMENTO DE RELACIONES EXTERIORES DE LOS EE UU Y GUINEA CORTARON SUS RELACIONES DIPLOMATICAS.

This story is particularly short and so took less than three CPU seconds to process. FRUMP understood that the diplomatic link from the U.S. to Guinea was ended, and it inferred that the link from Guinea to the U.S. was ended as well. The result of processing the article is a conceptual representation. Because the meaning representation is language free, it is easy to generate other natural languages as English.

The following story demonstrates FRUMP's ability to correctly classify the topic of the story. The sketchy script $NATIONALIZE is used to process the story despite the lack of any "key word" to indicate a nationalize situation. FRUMP never uses key words. Sketchy scripts are indexed by conceptualizations. FRUMP's solution to the script selection problem is too involved to be discussed here. See DeJong (1979) for a complete discussion of FRUMP's script selection methods.

INPUT:

TEHERAN, IRAN, MARCH 20 (UPI)-THE SHAH OF IRAN TODAY FORMALLY TOOK CONTROL OF THE COUNTRY'S MULTIBILLION DOLLAR OIL INDUSTRY FROM FOREIGN OPERATORS.

SHAH MOHAMMED REZA PAHLAVI ANNOUNCED NATIONALIZATION OF THE INDUSTRY IN HIS BROADCAST PERSIAN NEW YEAR MESSAGE. HE DECLARED THE TAKE-OVER GAVE IRAN "FULL AND REAL CONTROL" OF ALL OIL OPERATIONS.

THE ORDER PLACES UNDER CONTROL OF THE NATIONAL IRANIAN OIL COMPANY THE LARGEST OILFIELD, THE LARGEST REFINERY, THE LARGEST EXPORT TERMINAL AND THE LARGEST MAN-MADE ISLAND IN THE WORLD. THE FACILITIES HAD BEEN OPERATED BY A CONSORTIUM OF BRITISH, AMERICAN, FRENCH AND DUTCH CONCERNS. THE NATIONALIZATION CAME AFTER THE GROUP REFUSED TO DOUBLE IRANIAN OIL PRODUCTION TO EIGHT MILLION BARRELS PER DAY.

CPU TIME FOR UNDERSTANDING = 3866 MILLISECONDS

ENGLISH:
IRAN HAS NATIONALIZED ITS OIL INDUSTRY.

FRENCH:
L'IRAN A NATIONALISE SOCIETE PETROLIERE.

CHINESE:
ILAANG BAA I GE ILAANG SHYRYOU GONGYEH SHOUGUEIGWOYEOULE.

SPANISH:
IRAN NATIONALIZO SU INDUSTRIA PETROLERA.

INPUT: MOUNT VERNON, ILL, (UPI)—A SMALL EARTHQUAKE SHOOK SEVERAL SOUTHERN ILLINOIS COUNTIES MONDAY NIGHT, THE NATIONAL EARTHQUAKE INFORMATION SERVICE IN GOLDEN, COLO., REPORTED.

SPOKESMAN DON FINLEY SAID THE QUAKE MEASURED 3.2 ON THE RICHTER SCALE, "PROBABLY NOT ENOUGH TO DO ANY DAMAGE OR CAUSE ANY INJURIES." THE QUAKE OCCURRED ABOUT 7:48 P.M. CST AND WAS CENTERED ABOUT 30 MILES EAST OF MOUNT VERNON, FINLEY SAID. IT WAS FELT IN RICHLAND, CLAY, JASPER, EFFINGTON AND MARION COUNTIES.

SMALL EARTHQUAKES ARE COMMON IN THE AREA, FINLEY SAID.

SELECTED SKETCHY SCRIPT $EARTHQUAKE

CPU TIME FOR UNDERSTANDING = 3040 MILLISECONDS

ENGLISH SUMMARY:
THERE WAS AN EARTHQUAKE IN ILLINOIS WITH A 3.2 RICHTER SCALE READING.

This story took just over three CPU seconds to process and helps to illustrate why the entire story must be skimmed. The structure of news articles is such that often a FRUMP-like summary can be produced by simply parroting back the first sentence. But that is not always acceptable. Here the information about the strength of the earthquake would be lost. News articles are often written in a style different from most other texts. FRUMP was designed not as a news report processor but as a general text processor whose domain happened to be news reports. To demonstrate the general applicability of FRUMP's PREDICTOR/SUBSTANTIATOR

organization the program does not rely heavily on knowledge about the structure of news articles not shared by other texts.

8.0 CONCEPTUAL DEPENDENCY

All of FRUMP's processing is done in *conceptual dependency* (Schank, 1973). A conceptual dependency representation is made up of roles and role fillers. Each role indicates the part its filler plays in the event being represented. For example the conceptual dependency structure for the English sentence

John went to New York.

```
((ACTOR    ("JOHN")
 <=>       ("PTRANS")
 OBJECT    ("JOHN")
 TO        ("NEWYORK") ))
```

The form of a conceptual dependency structure is

((role 1 (role-filler 1) role 2 (role-filler 2) . . .))

Thus in the above conceptual dependency structure ACTOR, <=>, OBJECT and TO are roles and *JOHN*, *PTRANS*, *JOHN* and *NEWYORK* are their respective role fillers.

9.0 AN ANNOTATED EXAMPLE

The following example illustrates how the PREDICTOR and the SUBSTANTIATOR guide each other in the course of processing a one-sentence input text. The input text sentence is:

UGANDA TODAY TOOK FORMAL CONTROL OF AN AMERICAN OIL REFINERY.

Before this sentence was seen, the PREDICTOR had already predicted several conceptualizations. Among them is the following:

```
((<=>       ("ATRANS")
 MANNER     ("FORCED")
 ACTOR      ("POLITY")
 OBJECT     ("CONT")
    TYPE        ("ECONOMIC")
    PART        ("SPEC-INDUSTRY")
 TO         ("POLITY")
 FROM       ("POLITY") ))
```

This is a conceptual dependency representation for the event of one country taking economic control of a specific industry from another. It is this prediction that FRUMP must match with the input. On the left below is the computer output generated during processing. Each output is prefaced with the module from which it came. On the right are explanatory comments on the processing that resulted in the output message. It should be noted in reference to word numbers that the computer begins numbering the input words from zero.

Input:

UGANDA TODAY TOOK FORMAL CONTROL OF AN AMERICAN OIL REFINERY.

Computer Output	Comments
SUBSTANTIATOR: ((<=> (*ATRANS*) MANNER (*FORCED*))) BUILT FROM WORD (2) WORD SENSE TAKE1 PARTIALLY MATCHES A PREDICTION	SUBSTANTIATOR found the word "took" which has a word sense that partially matches several predicted conceptuolizations
PREDICTOR: PREDICTING ROLE (ACTOR) WILL BE FILLED WITH AN ELEMENT FROM LIST (*POLITY*)	PREDICTOR examined the partial conceptualization and predicted that the ACTOR role must be filled with a *POLITY*. The ACTOR of each predictions that could possibly be matched is filled with *POLITY*. Thus if this structure is going to match one of them, it must olso have a *POLITY* ACTOR.
SUBSTANTIATOR: PREDICTING (ACTOR) IS SUBJECT OF (TAKE1 2 NIL PAST)	Using its syntoctic knowledge, SUBSTANTIATOR determines that the ACTOR will probably be found os the subject of the verb "took".
FOUND POSSIBLE (*POLITY*) FROM WORD (0) UGANDA (ACTOR) HAS BEEN FILLED WITH (*UGANDA*) (TO) HAS BEEN FILLED WITH (*UGANDA*)	Indeed, a *POLITY* was found where the syntactic subject wos expected. Therefore it must be the conceptual ACTOR. The TO role is also filled with the same *POLITY* because the verb sense TAKE1 contains the information that its ACTOR and TO role fillers are the same.
PREDICTOR: PREDICTING ROLE (OBJECT) WILL BE FILLED WITH AN ELEMENT FROM LIST (*POSS* *CONT*)	There are several predicted conceptualizations that the portial one under construction can match. Some of them are abstroct transfers of POSSession, others of CONTrol. Thus, to differentiate which prediction the text might satisfy, PREDICTOR asks that the OBJECT be filled with either *POSS* or *CONT*.
SUBSTANTIATOR: PREDICTING (OBJECT) IS VOBJECT OF (TAKE1 2 NIL PAST)	SUBSTANTIATOR has used its syntactic knowledge to decide that if the conceptual OBJECT is specified in text it will be the object of the verb "took".

Computer Output	Comments
FOUND POSSIBLE (*ABSTRACT*) FROM WORD 4 (OBJECT) HAS BEEN FILLED WITH (*CONT*)	At word number 4, SUBSTANTIATOR found what it was looking for: a word that means *CONT*.
PREDICTOR: PREDICTING ROLE (OBJECT PART) WILL BE FILLED WITH AN ELEMENT FROM LIST (*HUMAN* *SPEC-INDUSTRY*)	Again PREDICTOR is trying to differentiate between several viable predictions. The (OBJECT PART) must be filled with either a human or a specific industry.
SUBSTANTIATOR: WORD (5) OF CAN POSSIBLY ADD (OBJECT PART) TENTATIVELY RESOLVING OF TO OF1	SUBSTANTIATOR found a preposition which it thinks can provide the desired information.
PREDICTING (OBJECT PART) IS POBJECT OF (OF1 5)	Here it is looking for the object of the preposition "of" at word 5.
FOUND POSSIBLE (*INDUSTRY*) FROM WORD 9 (OBJECT PART) HAS BEEN FILLED WITH (*REFINERY*)	As the object of the preposition SUBSTANTIATE found "refinery" which it knows is a kind of industry.
PREDICTING (OBJECT PART CLASS) IS MODIFIER OF (REFINERY1 9) FOUND POSSIBLE (*PRODUCT*) FROM WORD 8 (OBJECT PART CLASS) HAS BEEN FILLED WITH (*OIL*)	To be a specific industry, the kind of refinery must be determined. It decides that the kind of refinery, if present, will probably be an adjective modifier of "refinery" at word 9. It finds "OIL" at word 8.
PREDICTOR: PREDICTING ROLE (OBJECT TYPE) WILL BE FILLED WITH AN ELEMENT FROM LIST (*ECONOMIC*)	The PREDICTOR hos by now narrowed down the number of viable predicted conceptualizations to one. That one requires that the type of control taken over the industry be economic.
SUBSTANTIATOR: PREDICTING (OBJECT TYPE) IS MODIFIER LOOKING FOR MODIFIER OF (CONT1 4 10) TEXT ANALYZER UNABLE TO FIND MODIFIER	SUBSTANTIATOR decides that if the OBJECT TYPE role is present in the text it will be as an adjective modifier of word 4 "control" as in "took economic control." However, the input phrase does not say "economic control" so the text analyzer cannot add the OBJECT TYPE role.
TRYING INFERENCE RULE INFERENCER ASKS SLOT (OBJECT) BE	SUBSTANTIATOR decides to try to infer the desired role filler. It finds an inference rule that

Computer Output	Comments
FILLED WITH *CONT*	can add *ECONOMIC* in the OBJECT TYPE role of *ATRANS* acts provided certain conditions are met. Inference rules are indexed by the conceptual act, and the role they add. Thus they can be found efficiently. The conditions required by this rule include that control of an industry be changing hands. If that is true, then the control is probably of type *ECONOMIC*.
(OBJECT) ALREADY FILLED WITH *CONT*	
INFERENCER ASKS SLOT (OBJECT PART) BE FILLED WITH *INDUSTRY*	
(OBJECT PART) ALREADY FILLED WITH (*REFINERY*)	
ALL TESTS FOR INFERENCE ARE TRUE—INFERRING (OBJECT TYPE) IS (*ECONOMIC*)	
(OBJECT TYPE) HAS BEEN FILLED WITH (*ECONOMIC* CERTAINTY (7))	
PREDICTOR:	Finally PREDICTOR requests that the FROM role be filled with a *POLITY*.
PREDICTING ROLE (FROM) WILL BE FILLED WITH AN ELEMENT FROM LIST (*POLITY*)	
SUBSTANTIATOR:	However, SUBSTANTIATOR cannot add the FROM role using the text.
TEXT ANALYZER UNABLE TO ADD (FROM)—CALLING INFERENCE PROCEDURES	
TRYING INFERENCE RULE	An inference rule is found which says that for abstract transfers the entity giving up the object is probably the same as the current owner of the object. Thus the problem has been reduced to finding the OBJECT PART OWNER.
INFERENCER ASKS SLOT (OBJECT PART OWNER) BE FILLED WITH *POLITY*	
FILLER MISSING—SUBSTANTIATOR CALLED	SUBSTANTIATOR has decided that if the owner is specified in the text it is probably an adjective modifier of "refinery" at word 9. And indeed the owner is found to be the US.
PREDICTING (OBJECT PART OWNER) IS MODIFIER OF WORD (9)	
LOOKING FOR MODIFIER OF (REFINERY1 9)	
FOUND POSSIBLE (*ANIMATE*) FROM WORD 7	
(OBJECT PART OWNER) HAS BEEN FILLED WITH (*USA*)	
ALL TESTS FOR INFERENCE ARE TRUE—INFERRING (FROM) IS (*USA* CERTAINTY (9))	The inference is made.

Computer Output	Comments
PREDICTOR:	And finally the predicted conceptualization has
PREDICTED CONCEPTUALIZATION SATISFIED:	been fleshed out.

```
((<=>        (*ATRANS*)
 MANNER      (*FORCED*))
 ACTOR       (*UGANDA*)
 TO          (*UGANDA*)
 OBJECT      (*CONT*
    TYPE        (*ECONOMIC* CERTAINTY (7))
    PART        (*REFINERY*)
          TYPE       (*OIL*)
          OWNER      (*USA*)
 FROM        (*USA* CERTAINTY (9)) ))
```

The conceptualization produced contains the information that the industry changing hands is an oil refinery of the United States, that the country taking it is Uganda, and that the country giving it up is the United States. All of this was built in a very purposeful manner. The text was never examined without knowing what conceptual structure was to be built and approximately where in the text it would be found.

It seems as though a lot of work has been done to arrive at the correct parse of the sentence. Indeed, PREDICTOR and SUBSTANTIATOR each had to produce a large number of subresults. However, each of these subresults was achieved very efficiently. Very little work had to be done for any of them. The overall process is made much easier and more efficient because of the exchange of information between PREDICTOR and SUBSTANTIATOR.

10.0 A DAY IN THE LIFE OF FRUMP

On April 5, 1979 FRUMP was run continuously on UPI wire input for 24 hours. During this time 368 stories appeared on the wire. Of these, 100 were not actual news articles and 147 were not scriptal news articles. Thus there were 121 news stories that could in principle be understood by a script type approach such as FRUMP's. In fact, FRUMP had correct sketchy scripts for 29 of the articles of which 11 were processed correctly. On the average, 50 percent of the stories on the UPI wire are scriptal. FRUMP generally processes about 10 percent correctly. Here FRUMP correctly processed only about 3 percent of the total number of stories correctly. This is due to the comparatively small number of scriptal stories on April 5. Only about 33 percent of the total number of stories were scriptal

compared to the normal 50 percent. Furthermore, the distribution of scriptal situations was such that FRUMP's sketchy scripts accounted for less than one quarter of the scriptal stories. This figure is usually close to 30 percent. Taking these facts into account, FRUMP did quite well. It understood approximately 38 percent of the stories for which it had a sketchy script.

The following tables summarize the results from that day.

TABLE 1
Analysis of the FRUMP Run

Stories Understood Correctly	11
Stories Understood Almost Correctly (incorrect script varioble binding)	2
Stories Ignored (correct script present but not identified)	13
Stories Misunderstood (incorrect script used—correct script missing)	8
Stories Misunderstood (incorrect script used—correct script present)	1
Total Stories For Which Scripts Exist	29
Total Stories For Which Correct Script Was Selected	13

TABLE 2
Cause of Errors

Missing Vocabulary	11
Missing Script	9
Complex Syntax	2
Incorrectly Written Script	1
Problem Recognizing Name Group	1

Table 2 shows the cause of the errors. The primary difficulty is vocabulary. The second most important reason for errors is lack of the correct script. Seven of the stories which were incorrectly processed would have been processed correctly if FRUMP had the appropriate sketchy script. For these false alarms, FRUMP's vocabulary would probably have to be augmented as well. Two stories were missed because FRUMP's knowledge of syntax was insufficient. One story was missed because the writer of a sketchy script neglected to include relevant information. Another story processed wrong because FRUMP did not correctly identify the members of a group of people.

11.0 FRUMP'S KNOWLEDGE BASE

FRUMP currently has 48 sketchy scripts in its repertoir. They are:

$ACCEPT-BID	$ACCUSE	$AGREE
$APPROVAL	$ARREST	$ASSAULT
$ASYLUM	$BLOCKADE	$BREAK-NEGOTIATIONS
$BREAK RELATIONS	$COMMENT	$DEATH
$DEMAND	$DEMONSTRATION	$DENY
$DEPORT	$DETAIN	$EARTHQUAKE
$ELECTION	$ESTABLISH-NEGOTIATIONS	$ESTABLISH-RELATIONS
$EXPEL	$EXPLOSION	$FIGHTING
$FIND-ASSET	$GIVE-AID	$HOSPITAL
$KIDNAP	$MEET	$MOBILIZATION
$NATIONALIZE	$NO-AGREEMENT	$POSTPONE
$PRICE-INCREASE	$PROPOSE	$PROPOSE2
$PROTEST	$REDUCE-AID	$REJECT
$SEIZURE	$SPILL	$STORM
$STRIKE	$STRUCTURE-FIRE	$TERROR
$THREATEN	$VEHICLE-ACCIDENT	$WEAPONS-TEST

FRUMP's present vocabulary is approximately 1200 words. This is relatively small and it accounts for the most important reason for missing stories in the April 5 run. As table 2 shows, 11 stories were missed or incorrectly processed due to insufficient vocabulary. The number of stories correctly processed would have been doubled but for vocabulary problems.

There are no insurmountable problems associated with increasing FRUMP's vocabulary. The dictionary definition of a single word is quite simple and therefore uses very little storage. Furthermore, FRUMP's efficiency is not compromised by increased vocabulary. None of FRUMP's processing is sensitive to the size of its vocabulary. Thus the largest obstacle to greater success for the system is the effort required to add four or five thousand more vocabulary items.

12.0 CONCLUSION

FRUMP has been a very successful program. It often understands new input directly from the UPI news wire. This is in large part due to the robustness of FRUMP's PREDICTOR/SUBSTANTIATOR system organization. This organization performs better than previous natural language systems because previous systems made parsing a much harder problem than it actually is. Those systems required their parsers to analyze input text without the benefit of the knowledge in the rest of the understanding system. Even systems which used predictive understanders did not communicate those predictions to their parsers. Winograd was on the right track by allowing his parser and semantic interpretation rules to communi-