

PREDICTION AND SUBSTANTIATION:
TWO PROCESSES THAT COMPRISE UNDERSTANDING

Gerald DeJong
Yale AI Project
Box 2158 Yale Station
New Haven, CT 06520
U.S.A.

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. 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.

INTRODUCTION

In this paper I will argue for a new and different method of natural language analysis. This radical departure from previous systems is motivated by the fact that previous systems did not and could not handle novel real world input. Previous systems either worked only for the specific sentences they were carefully prepared for ([5], [8], [12], [17], [20], [21], [22]) or the domain of input sentences that could be handled was excessively and artificially constrained ([2], [3], [7]).

This paper presents an approach to natural language analysis which has resulted in a very robust system without imposing excessive constraints on the domain of the input text. As we will see, these advantages are directly traceable to the fact that the parser is heavily integrated into the rest of the understanding system.

A program has been written which embodies the theory of parsing. The program (FRUMP for Fast Reading Understanding and Memory Program) skims input text for important facts rather than

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 N00014-75-C-1111.

reading it for detail. FRUMP is often able 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. 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 radical restructuring of the understanding process 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. We estimate that about 30% of the articles from the news wire are understandable by a script processor such as FRUMP. FRUMP understands about a third of these. That is, about 10% of the wire stories. There are several reasons why FRUMP does not attain the theoretical limit of 30%. 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 not eventually achieve the 30% figure. However, a concerted effort to supply the missing information will be necessary. Such a concerted effort will have to wait while FRUMP is in its research stages.

DEFICIENCIES OF PREVIOUS SYSTEMS

Most previous natural language understanding programs have been made up of at least two separate subsystems ([5], [8], [17], [21], [22]). A parser subsystem analyzes the input natural language text into some intermediate form. The intermediate representations range from surface representations like syntactic parse trees ([19], [10]) to simplified English [12] to representations involving conceptual primitives [14]. 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.

Whatever the form of the intermediate representation, its purpose is always to isolate the inferencer from the capriciousness of natural language. The inference process can be made simpler and more efficient if the input to the inferencer always appears in some canonical form. The intermediate representation provides this canonical form. However, in all of these systems one gets the feeling that there are actually two parsers: the one everyone admits to having which produces the intermediate representation, and a second one hidden in the "inferencer" which parses the output of the first parser to decide what it really means.

There are several systems which do not separate parsing from semantic analysis. In Winograd's SHRDLU [21] parsing and semantic interpretation routines could call each other freely. However, the parsing was always done first. The major difference is that the above systems parsed a entire sentence at a time while SHRDLU parsed less than a sentence at a time. The SOPHIE system [2] analyzes English text by using a grammar which contains both syntactic and semantic rules. This "semantic grammar" approach [3] makes the system very robust. However, since semantic rules (which are domain specific) are intertwined with domain independent rules, the system cannot easily be extended to new domains. A new semantic grammar must be written to process text from a different domain. A similar approach was taken by Epstein and Walker [7] in constructing a natural language front end to a medical data base. The grammar rules contain domain specific knowledge. Again this has the effect of tying the system to one domain.

None of the above systems worked well on real world natural language input from different domains. Either the systems were too fragile

to process anything but the example inputs they were designed for, or they constrained the domain of inputs to the point that the system could not be readily extended to new input domains.

Making the parser responsive to the needs of the understander means that the component modules of the system must be constantly communicating. Furthermore, the text will be interpreted in a goal-directed manner. The parser will only try to build meaning structures that are desired by other modules of the system. Text will be interpreted a little bit at a time with a parser that has a very good idea of what it ought to produce, and only in a rich predictive context set up by the expectations of the understander. Furthermore, in such a system there is no need for an intermediate representation produced by a parser artificially separated from the rest of the system. Rather a natural language system as a whole will behave as one integrated process.

Recently inferencers have become more and more predictive in nature. Compare, for example, the MARGIE inferencer [13] to the SAM script applier [5]. Rieger's inferencer tried to generate all possible inferences from every input. SAM, on the other hand, had ready made strings of inferences for various situations. It looked for situations in which to apply its ready made inferences rather than wildly making every possible inference. A somewhat similar approach was taken by Charniak in his system Ms. Malaprop [4]. Rumelhart's work on schemas [15] is also in the same vein. The HEARSAY II system [9] uses a predictive hypothesize-and-test paradigm. The GUS system [1] is also predictive. It carries on a conversation about trips much as a travel agent would. It has built in general specifications of what it needs to find out and directs the conversation accordingly.

Thus, 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 "frame-like" systems illustrates this. While a frame [11] is a very broad concept, all frame-like systems have one thing in common: they are largely top down processors. That is, they predict characteristics of an input before it is seen.

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

communication between the inferencer and the parser extremely difficult.

FRUMP PARSING

FRUMP does not suffer from these problems. FRUMP is a natural language understander that skims newspaper articles on many different domains. The system has two modules: a PREDICTOR 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 relatively unconstrained. Predictions and communication are on a conceptual level. FRUMP uses a conceptual representation scheme called conceptual dependency [16].

The process of giving substance to the predicted characterizations is very purposeful. Text is analyzed and inference routines are called only in response to PREDICTOR's anticipations of events. Thus SUBSTANTIATOR can channel its processing. SUBSTANTIATOR tries to interpret the text in a way that matches one of the outstanding predictions. It also applies only those inference rules which might be able to help satisfy the predictions.

In FRUMP, text analysis is driven from PREDICTOR's predictions, not from the input. FRUMP does not have conventional parser which produces conceptualizations when it is presented with an input sentence. Rather, the input text is parsed only when PREDICTOR wants a specific piece of information and 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.

Communication between the two modules assures that they are both operating in the richest and most constrained context possible. When SUBSTANTIATOR verifies a prediction, it tells PREDICTOR the actual conceptualization it found. On the basis of the additional information included in the fleshed out conceptualization, 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 SUBSTANTIATOR to redirect its efforts. Thus the text analyzer and the inference processes of SUBSTANTIATOR are always operating in the most complete and constrained context that PREDICTOR

can provide.

THE PREDICTOR

The PREDICTOR's job is to make predictions about what events are likely to occur given the current context. The predictions can be on almost any conceptual level. That is, PREDICTOR can anticipate that particular a event is likely, or it can predict characteristics of a small part of an event, or it can predict groups of events. The prediction mechanism is centered on the idea of a sketchy script.

A sketchy script is a data structure that organizes FRUMP's 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 forty eight different situations ranging from earthquakes to countries establishing diplomatic ties to labor strikes. When FRUMP realizes it is reading a story about a particular situation, it applies the knowledge from the corresponding sketchy script in order to predict what events are likely to occur. FRUMP's understanding process includes constructing representations from the input text and inferring missing but implied events.

Scripts have been used before for natural language processing ([18], [5]). 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.

The following is an example of one of FRUMP's sketchy scripts. These are the facts that FRUMP knows about kidnappings. The facts in the sketchy script are represented in conceptual terms. The English meanings of the conceptual representations are given here.

- 1) The kidnappers will probably communicate a ransom demand to the family, company, or government of the person kidnapped.
- 2) The local police, FBI or other police agencies might be called in
- 3) The ransom may or may not be met
- 4) If the ransom is met, predict that the kidnapped person will probably be released but might continue to be held or be killed

- 5) If the ransom demand is not met, predict that the person will likely be held longer or killed but might be released
- 6) The kidnappers may or may not be apprehended
- 7) If the kidnappers are caught, predict a court case trying them for kidnapping

In addition to these facts, FRUMP knows that the important data in a kidnapping are the identity of the kidnappers, the identity of the kidnapped person, the identity of the group or person to whom the ransom demand was made, the nature of the ransom demand and which of the above predictions were found in or inferred from the text.

The identities of the participants in the sketchy script are script variables. In understanding a text, FRUMP both tries to find instances of the predicted facts and to bind the script variables to the identities given in the text.

PREDICTOR uses sketchy scripts to predict the likely events in a situation. When FRUMP has decided on a sketchy script to use in understanding a text, PREDICTOR asks SUBSTANTIATOR to produce the conceptual dependency representations that make up that sketchy script. That is, it predicts the important events in that situation as specified by the sketchy script.

SUBSTANTIATOR often produces only a part of a predicted conceptual representation as it did at the beginning of the example. When this happens, PREDICTOR individually predicts the missing pieces of the conceptualization. In this way it leads SUBSTANTIATOR through the process of building up a complete conceptualization.

Of course, before PREDICTOR can take advantage of the important events in a sketchy script, it must decide to use that sketchy script from among all of its sketchy scripts. This is a serious problem, but a fast efficient solution has been found. It is described in some detail in [6]. Space prohibits discussing the sketchy script selection algorithm here. Suffice it to say that any of FRUMP's 48 sketchy scripts can be selected efficiently. Top down predictions are still given to SUBSTANTIATOR during the selection process. The complexity of the selection algorithm proportional to the log of the number of scripts in the system.

THE SUBSTANTIATOR

There are three ways SUBSTANTIATOR can flesh out or verify a prediction. SUBSTANTIATOR is composed itself of four modules. One for each of the three substantiation methods and a module to choose which substantiation method to use for any given prediction.

The three modules that actually build the conceptual dependency structures are the conceptualization inferencer, the text analyzer and the conceptual dependency role inferencer. SUBSTANTIATOR's selection routine decides which of the structure building modules to try first in substantiating a given prediction. If the selected module fails, another may be tried.

The conceptualization inferencer is the simplest of the structure building modules. This module is able to infer entire conceptualizations at once. The inferences are always script-related. For example, in a story about countries establishing diplomatic relations, this module is able to infer that if the U.S. establishes diplomatic ties with China, then very likely China established similar diplomatic ties with the U.S. as well. This module has little to do with top-down parsing and so will not be discussed further.

The text analyzer builds only a piece of a conceptual structure at a time. When the text analyzer is asked to find a role filler, it tries to predict the sentence location of the desired word. Syntactic knowledge is used as heuristics to locate desired words. For example, the module knows where to find the subject of a verb, prepositional object, adjective modifiers, etc.

The conceptual dependency role inferencer also adds only a piece of a conceptual structure at a time. However, it uses information already understood to infer a missing conceptual role. The role inference rules are indexed by conceptual acts and within each act by the role it can add. Applicable inference rules can therefore be found quickly.

The SUBSTANTIATOR selection routine decides which of the modules will be used in satisfying a prediction. If the prediction is an entire conceptualization, it tries the conceptualization inferencer first. If that fails, it calls the text analyzer to find a part of the predicted conceptualization. Any successes are reported back to PREDICTOR so that it may use the information to revise its predictions.

If a prediction is of a role and its filler, SUBSTANTIATOR uses either the text analyzer or the role inferencer. The module used depends on how accurately each estimates it can fill the role. The parsing heuristics and inference rules are tagged with how well they can be expected to work. The selection routine chooses the module that can fill the missing role the most certainly. If the selected method cannot add the desired role filler, the less certain techniques are tried.

Now we can see how natural it is to divide the system into a prediction module and a verification module. As far as the PREDICTOR is concerned, there is no difference between the text analyzer and the role inferencer. PREDICTOR asks for a role to be filled in a certain way and is told later whether or not it could be done. Both the role inferencer and the text analyzer work on a small piece of a conceptualization at a time. Both are guided by the predictions made for the desired role filler. The only difference between the effect of text analysis and a role added by inference is that the text analyzer tends to be more certain of its result. Inferences, by their very nature, are at best good guesses. Thus the text analyzer is in a sense the inferencer of first resort. Since text analysis and role inferences are treated exactly the same, they belong in the same module.

AN EXAMPLE

INPUT:

THE CHILEAN GOVERNMENT HAS SEIZED OPERATIONAL AND FINANCIAL CONTROL OF THE EL TENIENTE MINING COMPANY, ONE OF THE THREE BIG COPPER ENTERPRISES HERE IN WHICH UNITED STATES COMPANIES HAVE INTERESTS. WHEN THE KENNECOTT COPPER COMPANY, THE OWNERS, SOLD A 51 PER CENT INTEREST IN THE COMPANY TO THE CHILEAN STATE COPPER CORPORATION IN 1967 IT RETAINED A CONTRACT TO MANAGE THE MINE. ROBERT HALDEMAN, EXECUTIVE VICE PRESIDENT OF EL TENIENTE, SAID THE CONTRACT HAD BEEN "IMPAIRED" BY THE LATEST GOVERNMENT ACTION. AFTER A MEETING WITH COMPANY OFFICIALS AT THE MINE SITE NEAR HERE, HOWEVER, HE SAID THAT HE HAD INSTRUCTED THEM TO COOPERATE WITH EIGHT ADMINISTRATORS THAT THE CHILEAN GOVERNMENT HAD APPOINTED TO CONTROL ALL ASPECTS OF THE COMPANY'S OPERATIONS.

SELECTED SKETCHY SCRIPT \$NATIONALIZE

CPU TIME FOR UNDERSTANDING = 3646 MILLISECONDS

ENGLISH:

CHILE HAS NATIONALIZED AN AMERICAN OWNED COMPANY.

FRENCH:

LE CHILI A NATIONALISE SOCIETE AMERICAINE.

CHINESE:

JYHLIH BAA I GE MEEIGWO GONGSY
SHOUGUEIGWOYEIOULE.

SPANISH:

CHILE NATIONALIZO UNA COMPANIA PREVIAMENTE
CONTROLADO POR INTERESES NORTE AMERICANOS.

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 parsing algorithm. FRUMP succeeds where previous natural language systems failed 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.

There are limitations to FRUMP's applicability. FRUMP can understand only situations for which it has a sketchy script, and there are some situations for which no sketchy script can be written. For example, FRUMP cannot understand editorials. Editorials are attempts to persuade the reader. This usually involves giving the reader arguments for adopting a particular view. However, the arguments are usually of a novel form. They often explain a heretofore unrealized consequence of adopting the right or wrong position. The sketchy script for a situation must include all of the important facts of that situation that FRUMP is to understand. There is no way to anticipate these novel, unforeseen ramifications at the time the sketchy script is written. Editorials by their very nature are not script-like and so cannot be processed by a script understander.

However, this does not argue that FRUMP's parsing techniques cannot be applied to a larger class of articles. It is a weakness of FRUMP's PREDICTOR module, not a breakdown in the PREDICTOR/SUBSTANTIATOR dialogue, that makes understanding these articles impossible for FRUMP. A more powerful PREDICTOR which can provide the SUBSTANTIATOR module with accurate characterizations of possible inputs for these

stories is needed. With a more powerful PRE-DICTOR, FRUMP-like parsing could be applied to a much larger class of text inputs.

REFERENCES

- [1] Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D. A., Thompson, H., and Winograd, T. (1977). GUS, a frame driven dialog system. Artificial Intelligence, Vol. 8, No. 1.
- [2] Brown, J. S., and Burton, R. R. (1975). Multiple representations of knowledge for tutorial reasoning. In D. Bobrow and A. Collins (Eds.), Representation and Understanding, Academic Press, New York.
- [3] Burton, R. R. (1976). Semantic grammar: an engineering technique for constructing natural language understanding systems. BBN Report 3453, ICAI Report 3. Bolt Beranek and Newman Inc., Boston.
- [4] Charniak, E. (1977). Ms. Malaprop, a language comprehension program. Proceedings of IJCAI-77, Cambridge, MA.
- [5] Cullingford, R. (1978). Script application: computer understanding of newspaper stories. Ph.D. Thesis, Yale University, New Haven, CT.
- [6] DeJong, G. (1979). Skimming stories in real time. Ph.D. Thesis, Yale University, New Haven, CT.
- [7] Epstein, M. and Walker, D. (1978). Natural language access to a melanoma data base. SRI Technical Note 171, Menlo Park, CA.
- [8] Kaplan, R. M. (1975). On process models for sentence analysis. In D. A. Norman and D. E. Rumelhart (eds.), Explorations in Cognition. W. H. Freeman and Company, San Francisco.
- [9] Lesser, V. and Erman, L. (1977). A retrospective view of the HEARSAY-II architecture. Proceedings of IJCAI-77, Cambridge, MA.
- [10] Marcus, M. (1977). A theory of syntactic recognition for natural language. Unpublished Ph.D. thesis, MIT, Cambridge, MA.
- [11] Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (ed.), The Psychology of Computer Vision. McGraw-Hill, New York, NY.
- [12] Parkinson, R., Colby, M., and Faught, W. (1976). Conversational language comprehension using integrated pattern-matching and limited parsing. Technical Report, UCLA Psychiatry Department, Los Angeles, CA.
- [13] Rieger, C. (1974). Conceptual memory. Ph.D. Thesis, Computer Science Department, Stanford University, Stanford, CA.
- [14] Riesbeck, C. K. (1975). Conceptual analysis. In R. Schank (ed.), Conceptual Information Processing. North Holland, Amsterdam.
- [15] Rumelhart, D. (1975). Notes on a schema for stories. In D. Bobrow and A. Collins (eds.), Representation and Understanding. Academic Press, New York.
- [16] Schank, R. C. (1973). Identification of conceptualizations underlying natural language. In R. C. Schank, and K. Colby (eds.), Computer Models of Thought and Language. W. H. Freeman and Co., San Francisco.
- [17] Schank, R. C. (1975). Conceptual Information Processing. North Holland, Amsterdam.
- [18] Schank, R. C. and Abelson, R. P. (1977). Scripts, Plans, Goals, and Understanding. Lawrence Erlbaum Press, Hillsdale, NJ.
- [19] Thorne, J., Bratley, P., and Dewar, H. (1968). The syntactic analysis of English by machine. In D. Michie (ed.), Machine Intelligence 3, American Elsevier Publishing Company, New York.
- [20] Wilks, Y. (1973). An artificial intelligence approach to machine translation. In R. C. Schank and K. Colby (eds.), Computer Models of Thought and Language. W. H. Freeman and Co., San Francisco.
- [21] Winograd, T. (1972). Understanding Natural Language. Academic Press, New York.
- [22] Woods, W. A. (1970). Transition network grammars for natural language analysis. Communications of the ACM. Vol. 13, p. 591.