

IDIOLECTIC LANGUAGE-ANALYSIS FOR
UNDERSTANDING DOCTOR-PATIENT DIALOGUES*

Horace Enea and Kenneth Mark Colby

Department of Computer Science
Stanford University
Stanford, California

ABSTRACT

A programming language is described which is designed to simplify the construction of computer programs to analyze English. This system attempts to merge the best features of pattern matchers and the phrase structure approach to language analysis. Several practical problems which occur in dealing with such a system are described.

INTRODUCTION

Why is it so difficult for machines to understand natural language? Perhaps it is because machines do not simulate sufficiently what humans do when humans process language. Several years of experience with computer science and linguistic approaches have taught us the scope and limitations of syntactic and semantic parsers. (Schank, Tesler and Weber,⁸ Simmons,⁹ Winograd,¹³ Woods¹⁴). While extant linguistic parsers perform satisfactorily with carefully edited text sentences or with small dictionaries, they are unable to deal with everyday language behavior characteristic of human conversation. In a rationalistic quest for certainty and attracted by an analogy from the proof theory of logicians in which provability implied computability, computational linguists hoped to develop formalisms for natural language grammars. But the hope has not been realized and perhaps in principle cannot be. (It is difficult to formalize something which can hardly be formulated).

Linguistic parsers use morphographemic analyses, parts-of-speech assignments and dictionaries containing

multiple word-senses each possessing semantic features, programs or rules for restricting word combinations. Such parsers perform a detailed analysis of every word, valiantly disambiguating at each step in an attempt to construct a meaningful interpretation. While it may be sophisticated computationally, a conventional parser is quite at a loss to deal with the caprice of ordinary conversation. In everyday discourse people speak colloquially and idiomatically using all sorts of pat phrases, slang and cliches. The number of special-case expressions is indefinitely large. Humans are cryptic and elliptic. They lard even their written expressions with meaningless fillers and fragments. They convey their intentions and ideas in idiosyncratic and metaphorical ways, blithely violating rules of 'correct' grammar and syntax. Given these difficulties, how is it that people carry on conversations easily most of the time while machines thus far have found it extremely difficult to continue to make appropriate replies indicating some degree of understanding?

It seems that people 'get the message' without always analyzing every single word in the input. They even ignore some of its terms. People make individualistic and idiosyncratic selections from highly redundant and repetitious communications. These personal selective operations, based on idiosyncratic intentions, produce a transformation of the input by destroying and even distorting information. In speed reading, for example, only a small percentage of contentive words on each page need be looked at. These words somehow resonate with the readers relevant conceptual-belief structure whose processes enable him to 'understand' not simply the language but all sorts of unmentioned aspects about the situations and events being referred to in the language. Normal written English text is estimated to be 5/6 redundant (Rubenstein and Haberstroh⁷). Spoken conversations in English are probably better than 50% redundant (Carroll¹). Words can be garbled and listeners nonetheless get the gist or drift of what is being said. They see the "pattern" and thus can supply much of what is missing

To approximate such human achievements we require a new perspective and a practical method which differs from that of current linguistic approaches. This alternate approach should incorporate those aspects of parsers which have been found to work well, e.g., detecting embedded clauses. Also individualistic features characteristic of an idiolect should have dominant emphasis. Parsers represent complex and refined algorithms. While on one hand they subject a sentence to a detailed and sometimes overkilling analysis, on the other, they are finicky and oversensitive. A conventional parser may simply halt if a

* This research is supported by Grant PHS MH 06645-12 from the National Institute of Mental Health, by (in part) Research Scientist Award (No. 1-K05-K14,433) from the National Institute of Mental Health to the second author and (in part) by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-183).

word in the input sentence is not present in its dictionary. It finds ungrammatical expressions such as double prepositions ('Do you want to get out of from the hospital?') quite confusing. Parsers constitute a tight conjunction of tests rather than a loose disjunction. As more and more tests are added to the conjunction, the parser behaves like a finer and finer filter which makes it increasingly difficult for an expression to pass through. Parsers do not allow for the exclusions typical of everyday human dialogues.

Finally, it is difficult to keep consistent a dictionary of over 500 multiple-sense words classified by binary semantic features or rules. For example, suppose a noun (Ni) is used by some verbs as a direct object in the semantic sense of a physical object. Then it is noticed that Ni is also used by other verbs in the sense of a location so 'location' is added to Ni's list of semantic features. Now Ni suddenly qualifies as a direct object for a lot of other verbs. But some of the resultant combinations make no sense even in an idiolect. If a special feature is then created for Ni, then one loses the power of general classes of semantic features. Adding a single semantic feature can result in the propagation of hidden inconsistencies and unwanted side-effect.. as the dictionary grows it becomes increasingly unstable and difficult to control.

Early attempts to develop a pattern-matching approach using special-purpose heuristics have been described by Colby, Watt and Gilbert,² Weizenbaum¹¹ and Colby and Enea.³ The limitations of these attempts are well known to workers in artificial intelligence. The man-machine conversations of such programs soon becomes impoverished and boring. Such primitive context-restricted programs often grasp a topic well enough but too often do not understand quite what is being said about the topic, with amusing or disastrous consequences. This shortcoming is a consequence of the limitations of a pattern-matching approach lacking a rich conceptual structure into which the pattern abstracted from the input can be matched for inferencing. These programs also lack a subroutine structure, both pattern directed and specific, desirable for generalizations and further analysis.

The strength of these pattern matching approaches lies in their ability to ignore some of the input. They look for patterns, which means the emphasis of some detail to the exclusion of other detail. Thus they can get something out of nearly every sentence-- sometimes more, sometimes less.

An interesting pattern-matching approach for machine translation has been developed by Wilks.¹² His program constructs a pattern from English text input which is matched against templates in an interlingual data base from which, in turn, French output is generated without using a generative grammar.

In the course of constructing an interactive simulation of paranoia we were faced with the problem of dealing with unedited and unrestricted natural language as it is used in the doctor-patient situation of a psychiatric

interview.(Colby, Hilf, Weber, and Kraemer,⁴ Colby and Hilf⁵). This domain of discourse admittedly contains many psychiatrically stereotyped expressions and is constrained in topics (Newton's laws are rarely discussed). But it is rich enough in verbal behavior to be a challenge to a language understanding algorithm since a variety of human experiences are discussed domain including the interpersonal relation which develops between the interview participants. A look at the contents of a thesaurus reveals that words relating to people and their interrelations make up roughly 70% of English words.

The diagnosis of paranoia is made by psychiatrists relying mainly on the verbal behavior of the interviewed patient. If a paranoid model is to exhibit paranoid behavior in a psychiatric interview, it must be capable of handling dialogues typical of the doctor-patient context. Since the model can communicate only through teletyped messages, the vis-a-vis aspects of the usual psychiatric interview are absent. Therefore the model must be able to deal with unedited typewritten natural language input and to output replies which are indicative of an underlying paranoid thought process during the episode of a psychiatric interview.

In an interview there is always a who saying something to a whom with definite intentions and expectations. There are two situations to be taken into account, the one being talked about and the one the participants are in. Sometimes the latter becomes the former. Participants in dialogues have intentions and dialogue algorithms must take this into account. The doctor's intention is to gather certain kinds of information while the patient's intention is to give information and get help. A job is to be done; it is not small talk. Our working hypothesis is that each participant in the dialogue understands the other by matching selected idiosyncratically- significant patterns in the input against conceptual patterns which contain information about the situation or event being described linguistically. This understanding is communicated reciprocally by linguistic responses judged appropriate to the intentions and expectations of the participants and to the requirements of the situation. In this paper we shall describe only our current input-analyzing processes used to extract a pattern from natural language input. In a later communication we shall describe the inferential processes carried out at the conceptual level once a pattern has been received by memory from the input-analysing processes.

Studies of our 1971 model of paranoia (PARRY) indicated that about thirty percent of the sentences were not understood at all, that is, no concept in the sentence was recognized. In a somewhat larger number of cases some concepts, but not all, were recognized. In many cases these partially recognized sentences lead to a partial understanding that was sufficient to gather the intention of the speaker and thus lead to output an appropriate response. However, misunderstandings occurred too often. For example:

DOCTOR: How old is your mother ?

PARRY: Twenty-eight

PARRY has interpreted the question as referring to his own age and answered by giving his age. The purpose of our new language analysis system is to significantly raise the level of understanding by preventing such misunderstandings while not restricting what can be said to PARRY. We do not expect complete understanding from this system -- even native speakers of the language do not completely understand the language.

By 'understanding' we mean the system can do some or all of the following:

- 1) Determine the intention of the interviewer in making a particular utterance.
- 2) Make common logical deductions that follow from the interviewers utterance
- 3) Form an idiolectic internal representation of the utterance so that questions may be answered, commands carried out, or data added to memory.
- 4) Determine references for pronouns, and other anaphora.
- 5) Deduce the tone of the utterance, i.e., hostile, insulting...
- 6) Classify the input as a question, rejoinder, command, ...

The approach we are taking consists of merging the best features of pattern directed systems such as the MAD DOCTOR,² ELIZA¹¹ and parsing directed systems for example, Winograd,¹³ Woods.¹⁴ By merging the BNF phrase structure approach of analyzing English with the pattern matching approach, with its attendant emphasis of some concepts to the exclusion of others. The programs to accomplish this are written in MLISP2, an extensible version of the programming language MLISP,^{6,10} and uses an interpreted version of the pattern matcher designed for a new programming language LISP70.

The following is a basic description of the pattern matcher. We shall illustrate its operation using examples of problems common to teletyped psychiatric dialogues.

PATTERN MATCHING

Pattern directed computation involves two kind of operations on data structures: decomposition and recomposition. Decomposition breaks down an input stream into components under the direction of a decomposition pattern

("dec"). The inverse operation, recomposition, constructs an output stream under the direction of a recomposition pattern ("rec").

A rewrite rule is of the form:

dec → rec

It defines a partial function on streams as follows: if the input stream matches the dec, then the output stream is generated by the rec. The following rule (given as an example only) could be part of a question answering function:

How are you ? → Very well and you ?

If the input stream consists of the four tokens:

How are you ?

the output stream will consist of the five tokens:

Very well and you ?

REWRITE FUNCTIONS

A rewrite rule defines a partial function, for example, the mapping of some particular token into some other particular token. A broader partial function can be defined as the union of several rewrite rules. A rewrite function definition is of the form:

```
RULES OF <name> =  
  dec1 → rec1,  
  dec2 → rec2,  
  ...  
  decn → recn;
```

VARIABLES

A function is difficult to define if every case must be enumerated. Therefore, rewrite rules allow variables to appear in patterns. The value of a variable can be either a list or an atom. In this paper the notation:

:X

where X is any identifier, will denote the variable X. The variables of each rule are distinct from the variables of all other rules, even if their names are the same.

The following definition has only three rewrite rules, but handles an unlimited number of input streams:

RULES OF REPLY=

HOW ARE YOU '?' → VERY WELL '?

HOW IS :X → I HAVEN'T SEEN :X ',
LATELY'.

DID :X GO TO :Y '?' →

WHY DON'T YOU ASK :X YOURSELF'?

A variable can appear more than once in a single dec pattern, but it must match identical items at each appearance. Example:

RULES OF EQUAL =

(EQUAL :X :X) → TRUE;

ELLIPSIS

To make patterns easier to read and write, the ellipsis symbol ... can be used to stand for an unnamed variable. Thus:

IS ... COMING → NO, ... COULD NOT MAKE IT.

If an ellipsis (...) occurs several times on a side, it designates a different variable each time. The n'th ellipsis in a dec designates the same variable as the n'th ellipsis in the rec.

Ellipsis is one of the principle ideas of pattern matching. It permits imprecise matching; that is, the emphasis or ignoring of items.

AUTOMATIC ORDERING OF RULES

The order of rules in a function definition does not specify the order in which the system will attempt to apply them. This ordering operation is handled by a special system ordering function. Consider the rewrite function:

RULES OF REPLY =

I SEE :X → SO WHAT '?,

I SEE ANN → WOW '!;

Both rules would match:

I SEE ANN

In such cases the more specific rule takes precedence. Thus given:

I SEE ANN

as the input stream, the output stream would be:

WOW !

but given:

I SEE STARS

the output stream would be:

SO WHAT ?

A literal is more specific than a variable. A variable appearing for the second time is more specific than a variable appearing for the first time in a dec. This is so because the second occurrence of the variable must match the same pattern as the first occurrence. The precedence function is itself written in rewrites and so is both extendable and changable by the user. Currently precedence is calculated by a left to right application of the above criteria. Therefore, the following function defines the LISP function EQUAL:

RULES OF EQUAL =

(EQUAL :X :X) → T,

(EQUAL :X :Y) → NIL;

SEGMENTS

Sometimes it is desirable for a variable to match an indeterminate number of items. This is notated:

::X

Use of the double-colon ("::") means that the variable (e.g., X) will match zero or more items. Example:

RULES OF APPEND=

(APPEND (::X) (::Y)) → (::X ::Y);

or if the input stream were:

(APPEND (A B) (C D E))

the output stream would be:

(A B C D E)

For increased readability the rule could also be written:

RULES OF APPEND =

(APPEND (...) (...)) → (... ...);

Another example:

RULES OF REPLY =

WHERE DID ::X GO →

::X WENT HOME '.,

Therefore,

WHERE DID THE CARPENTER GO →

THE CARPENTER WENT HOME.

APPLICATION

One of the main deficiencies of the system in which the MAD DOCTOR was programmed was its lack of adequate subroutining capability. Subroutines may be indicated in the rewrite system as follows:

```

RULES OF LAST =
  () → (),
  (:X) → :X,
  (:X ...) → <LAST (...)>;

```

The "<>" surrounding a pattern means that the current input stream is to be pushed down, that the function indicated by the first token within the brackets is to be entered with the rest of the pattern appended to the front of the input stream, and that the output stream is to be placed into the restored current input stream. Note that MLISP2 functions may be called as well as rewrite functions.

GOALS

To gain the advantage of goal directed pattern matching and computing, as well as the full power of context sensitive grammars, the following form may be used:

```

RULES OF PREPOSITIONAL_PHRASE =
  <PREPOSITION>:P <NOUN_PHRASE>:N
  → (PREP_PH :P :N);

```

The identifier between the angled brackets ("<>") names a rewrite function the rules of which are to be matched against the input stream. When a match occurs the output stream of the goal will be bound to the associated variable. Example:

```

RULES OF PREPOSITIONAL_PHRASE =
  <PREPOSITION>:P <NOUN_PHRASE>:N
  → (PREP_PH :P :N);

```

```

RULES OF NOUN_PHRASE =
  TOWN → (NOUN_PH TOWN),
  PALO ALTO → (NOUN_PH PALO_ALTO);

```

```

RULES OF PREPOSITION =
  IN → IN,
  ON → ON;

```

and the input stream:

```
IN PALO ALTO
```

the output stream would be:

```
(PREP_PH IN (NOUN_PH PALO_ALTO))
```

OPTIONALS

Many other shorthands exist to simplify writing rules. One useful feature that will be mentioned here is the optional.

```

RULES OF AUXILIARY_PHRASE =
  <AUXILIARY>:A [<NEGATIVE>:N1:N1 →
  (AUX_PH :A [:N1:N1)];

```

If the optional pattern, enclosed in square brackets ("[]"),

occurs in the input stream it will be bound to :N. :N1 will be bound to 2. If the <NEGATIVE> does not occur, :N1 will be bound to 1. On the rec side of the rules if :N1 is 2 then :N will be placed in the output stream. If it is 1 then nothing is placed in the output stream at that point. Example, given the rule above:

```

DO → (AUX_PH DO)
DO NOT → (AUX_PH DO NOT)

```

MORE EXAMPLES

We have collected a large number of dialogues using our previous program PARRY. These dialogues form a large body of examples of the kind of English which we can expect. Martin Frost, a graduate student in Computer Science, Stanford University, has written a keyword in context program which enables us to isolate examples centered on particular words so that uses of those words in context become more apparent. Our general approach is to build a system which can produce desired interpretations from these examples and to incrementally add to the rules in the system as new cases are discovered during the running of the program.

Following are some examples of commonly occurring situations and examples of the kind of rules we use to handle them.

QUESTION INTRODUCER

In doctor-patient dialogues it is quite common to introduce a question by the use of a command. The "question introducer" is followed by either a <NOUN_PHRASE> or a <DECLARATIVE_SENTENCE>. For example,

```
COULD YOU TELL ME YOUR NAME?
```

Rather than attempt a literal analysis of this question, which might lead to the interpretation:

```
DO YOU HAVE THE ABILITY TO SPEAK YOUR NAME TO ME?
```

we utilize rules like:

```

RULES OF SENTENCE =
  <QUESTION_INTRODUCER>:Q <NOUN_PHRASE>:N
  → (IS :N '*?'* );

```

```

RULES OF QUESTION_INTRODUCER =
  COULD YOU TELL ME → ,
  WOULD YOU TELL ME → ,
  PLEASE TELL ME → ;

```

Although it is conceivable that there are an infinite number of ways to introduce a question in this manner, we have found only about six literal strings are actually used in our data base of dialogues. When we discover a new string we incrementally add a rule. When we have enough examples

to detect a more general form we replace the rules for <QUESTION_INTRODUCER> by a more elegant and general formulation. This approach allows us to process dialogues before we have a complete analysis of all possible sentence constructions, and it allows us to build a language analyzer based on actually occurring forms.

Notice that it is possible to make more than one analysis of any given sentence depending on what is being looked for. A poet might be interested in the number of syllables per word and the patterns of stress. A "full" analysis of English must allow for this possibility, but it is clearly foolish to produce this kind of analysis for PARRY. Our analysis will be partial and idiosyncratic to the needs of our program. This is what is meant by idiolectic.

FILLERS

It is quite common for interviewers to introduce words of little significance to PARRY into the sentence. For example:

WELL, WHAT IS YOUR NAME?

The "well" in this sentence serves no purpose in PARRY's analysis, although it might to a linguist interested in hesitation phenomena. These fillers can be ignored. The following rules accomplish this:

```
RULES OF SENTENCE =
  <FILLERS>:F <SENTENCE>:S → :S;
```

```
RULES OF FILLERS =
  WELL → ,
  OK → ;
```

PUNCTUATION

Interviewers use little intra-sentence punctuation in talking to PARRY. When it is used it is often to separate phrases that might otherwise be ambiguous. Example:

WHY WEREN'T YOU VERY CLOSE, FRANK

Here the comma clearly puts "CLOSE" in a different phrase from "FRANK". Punctuation, when used in PARRY's rules, is generally enclosed in optional brackets ("[]"). This has the effect of separating phrases when punctuation is used, but not requiring full punctuation for the system to work. Example:

```
RULES OF SENTENCE =
  <SENTENCE>:S1 [' , ]:C <SENTENCE_CONNECTOR>:SC
  <SENTENCE>:S2
  → (CONJUNCTION :SC :S1 :S2);
```

CLICHES AND IDIOMS

The English we encounter in doctor-patient

dialogues is made up of a great number of clichés and idioms, therefore we anticipate a large number of rules devoted to them. For example:

```
RULES OF TIME_PHRASES =
  A COUPLE OF <TIME_UNIT>:T AGO
  → (TIME (RELATIVE PAST) (REF PRESENT) :T);
```

```
RULES OF TIME_UNIT =
  SECONDS → (WITHIN CONVERSATION),
  MOMENTS → (WITHIN CONVERSATION),
  DAYS → (BEFORE CONVERSATION DAYS);
```

REPRESENTATION CORRECTION

Intermediate results are often produced which are misleading in meaning or are in the wrong form for further processing. We, therefore, incorporate at various points rules which detect certain undesired intermediate results and convert them to the desired form. Example:

```
RULES OF CORRECT_FORM =
  (QUESTION ... (SENTENCE ...)) →
  (QUESTION ... ...);
```

UNKNOWN WORDS

Rules can be derived to handle words which were previously unknown to the system. For example:

```
RULES OF UNKNOWN_WORD =
  DR'. :X → <NEW_WORD NAME :X>,
  THE :X <VERB_PHRASE>:V →
    <NEW_WORD NOUN :X>,
  I :X YOU → <NEW_WORD VERB :X>;
```

Here "NEW_WORD" is a function which adds new words to the dictionary.

CONCLUSION

We are faced with the problems of natural language being used to interview people in a doctor-patient context. We have developed a language processing system which we believe is capable of performing in these interviews at a significantly improved level of performance compared to systems used in the past. We have developed techniques which can measure performance in comparison with the ideal of a real human patient in the same context.^{4,5,7} We are designing our system with the realization that a long period of development is necessary to reach desired levels of performance. This is a system that can work at a measured level of performance and be improved over time with new rules having minimum interaction with those already existing. Our system is designed so that a complete analysis of every word or phrase of an utterance is not necessary.

The basis of this system is a rewrite interpreter which will automatically merge new rules into the set of

already existing rules so that the system will continue to handle sentences which it handled in the past.

REFERENCES

- ¹Carroll, J.B. Language and Thought. Prentice-Hall, Englewood Cliffs, New Jersey, p. 59.
- ²Colby, K.M., Watt, J. and Gilbert, J.P. A computer method of psychotherapy. Journal of Nervous and Mental Disease, 142,148-152,1966.
- ³Colby,K.M. and Enea,H. Heuristic methods for computer understanding of natural language in context restricted on-line dialogues. Mathematical Biosciences,1,1-25,1967.
- ⁴Colby, K.M., Hilf, F.D., Weber, S., and Kraener, H. Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes. Artificial Intelligence,3,199-221,1972.
- ⁵Colby, K.M. and Hilf, F.D. Multidimensional analysis in evaluating the adequacy of a simulation of paparnoid processes. Memo AIM-194. Stanford Artificial Intelligence project, Stanford University.
- ⁶Enea, H. MLISP, Technical report no. CS-92, 1968, Computer Science Department, Stanford University.
- ⁷Rubenstein, A.H. and Haberstroh, C. J., Some Theories of Organization, Dorsey Press, Homewood Ill.,1960, p. 232.
- ⁸Schank, R.C., Tesler, L. and Weber,S. Spinoza ii: Conceptual case-based natural language analysis. Memo AIM-109, 1970, Stanford Artificial Intelligence Project, Stanford University.
- ⁹Simmons, R.F. Some semantic structures for representing English meanings. Preprint, 1970, Computer Science Department, University of Texas, Austin.
- ¹⁰Smith, D.C., MLISP, Memo AIM-135, 1970, Stanford Artificial Intelligence Project, Stanford University.
- ¹¹Weizenbaum, J. Eliza- a computer program for the study of natural communication between man and machine. Communications of the ACM, 9,36-45,1966.
- ¹²Wilks, Y.A. Understanding without proofs. (See this volume).
- ¹³Winograd, T. A program for understanding natural language. Cognitive Psychology,3,1-191,1972.
- ¹⁴Woods, W.A. Transition network grammars for natural language analysis. Communications of the ACM,13,591-606,1970.