

AUTOMATIC QUESTION GENERATION FROM TEXT - AN AID TO INDEPENDENT STUDY¹

John H. Wolfe

Navy Personnel Research and Development Center
San Diego, California 92152

INTRODUCTION

This report describes an experimental computer-based educational system called automatic question generation (AUTOQUEST) for assisting independent study of written text. Studies of reading comprehension have shown that retention of material is enhanced if the student is periodically required to answer questions about what he has read (Anderson & Biddle, 1975; Anderson et al., 1974; Alessi et al., 1974; Anderson et al., 1975a, 1975b). This principle has been employed in computer-managed instruction, but it requires considerable human effort to prepare the questions.

The goal of AUTOQUEST is to automatically generate questions from text in order to improve independent study of any textual material. The AUTOQUEST system presents text on a computer terminal to a student, a paragraph at a time, and asks him questions about it, based on a randomly selected sentence contained in the paragraph. If the student's answer contains a certain percentage of the words in the original sentence, the student is told his answer is correct and he goes on to the next paragraph. If the answer is judged wrong, the paragraph is displayed again and another question is generated.

The research was almost entirely directed toward the development of the techniques for generating questions. Issues of economy or efficiency of programs did not receive much attention. Nor was the project concerned with running subjects to determine the pedagogical effectiveness of the system when compared with unsupervised study or conventional CAI. The project was devoted toward determining the feasibility of automatically producing a man-machine dialogue by natural language processing techniques with minimal preprocessing of text by human beings.

One of the first approaches to natural language processing was Weizenbaum's Eliza program (Weizenbaum, 1966), in which the computer simulated a nondirective psychotherapist. The program uses pattern matching keyed to certain words in the patient's

conversation with no real understanding of the content. For example, if the patient said, "I am very unhappy these days," the computer would notice the words "I am" and generate "How long have you...", followed by the remainder of the patient's statement so as to produce the question, "How long have you been very unhappy these days?" A more sophisticated pattern-matching system to simulate paranoia was developed by the psychiatrist Colby (Colby, Parkinson, and Frought, 1974). For reasons to be described later in this report, the pattern-matching approach is the one used by the AUTOQUEST system.

APPROACH

The approach to automatic question generation was entirely syntactic rather than semantic. That is, only the form of a sentence is treated, not its meaning. In this manner, a general system could be developed which would work on any body of English text, regardless of subject matter. If a semantic approach had been employed, the project would have been restricted to one or two specialized subjects and a great amount of effort would have been spent in developing semantic models for those subjects.

A pure syntactic approach has many limitations, of course. First, the student is required to give verbatim parts of the original text in his answers. Second, it is well known that many English sentences are syntactically ambiguous and can only be parsed correctly when their meaning is taken into account. It was our hope (justified later by our results) that these problems would not occur so frequently that a useful system would be infeasible.

¹This research was supported jointly by the Navy Personnel Research and Development Center and by the Advanced Research Projects Agency of the Department of Defense. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

The project's first attempt at syntactic question generation was a two-stage process in which a text sentence was put through a parser and the parsed sentence reassembled into the form of a question. For the parsing, we used a version of Woods' Augmented Transition Network Parser which was available on a computer at the University of California (UC) at Irvine. It was a slightly scaled-down version of his experimental Lunar Sciences Natural Language system with the so-called NASA grammar. The system is one of the most successful natural-language programs ever developed. It proved to be extremely useful in helping geologists retrieve information about lunar rocks. Unfortunately, a number of problems were encountered in adapting it to this work. The parsing program occupied so much computer memory that it could only be run after midnight at UC Irvine. Further, 50% of the text sentences failed to parse and gave error messages indicating they needed more memory. Of the remaining 50% parsed sentences, only 60% appeared to have been parsed correctly. As a check, one of the sentences that failed at UC Irvine was run at Bolt, Beranek, and Newman, Inc. (BBN) with the original version of the parser. The program ground away for over an hour before returning a message that space requirements had been exceeded.

This experience was sufficiently discouraging that it was decided not to try to adapt any existing general-purpose parser to this project. Instead, the decision was made to develop a specialized pattern-matching program that compares a sentence against a table of pre-stored patterns. If the sentence fits a pattern, a certain kind of question is generated from it. If the sentence does not fit any pattern in the table, it is ignored.

The following sentence, taken from a computer programming manual, helps to illustrate the process: "The dd name identifies a DD statement so that subsequent control statements and the data control block in the processing program can refer to it." The sentence matches a pre-stored pattern in the program of the form: "S₁ so that S₂."

The first part of the sentence, S₁, is scanned to locate the verb. If the first verb found is an auxiliary, such as "is," "was," "were," "do," etc., it is moved to the front of S₁. Otherwise, the tense and number of the verb are examined and an appropriate auxiliary created at the front of S₁. The transformed S₁ is called QFORM(S₁). The generated question is: "Why QFORM(S₁)?" The computer also generates an expected answer: "So that S₂." Applying these rules to the text example, we get the question, "Why does the dd name identify a DD statement?" with the expected answer, "So that subsequent control statements and the data control block in the processing program can refer to it."

The student's answer is checked to see if more than 50% of the long words (more than

4 letters) of the expected answer are contained in his answer. In this example, he has to come up with at least four of the words in the list (subsequent control statements block processing program) in order to have his answer judged correct. The 50% criterion allows the student some flexibility beyond a complete verbatim requirement. The restriction to long words was designed to eliminate common English words and increase the percentage of content-specific words.

If the expected answer contains a conjunction, such as "and" or "or," then the student's answer will be judged partly right if it is correct for part of the answer on either side of the conjunction. For example, if the student replied, "So that subsequent control statements can refer to it," the computer would come back saying, "Yes, subsequent control statements and the data control block in the processing program can refer to it. Your answer is partly right."

DESCRIPTION OF METHODS

The Question Generation Algorithms

Vocabulary. The program uses a dictionary of articles, pronouns, prepositions, conjunctions, and about 1700 verb stems. A morphological analysis subroutine strips off the endings of a word such as -ing, -s, -ed, -es and tests if the remainder of the word is in the verb list. The verbs are classified as transitive or intransitive, regular or irregular, and verb, verb-noun, verb-adjective, or verb-adjective-noun. For example, the word "control" is a verb-noun since it can be used either as a verb or as a noun; it is also transitive and regular. The word "bring" is always a transitive irregular verb.

A list of common adverbs also appears in the vocabulary. In addition, words ending in -ly which had not been assigned another part of speech were classified as adverbs.

Since nouns can modify nouns, as in the phrase, "job control language," no distinction was made between adjectives and nouns in the lexicon. Any words which were not in the vocabulary were automatically classified as nouns whenever they appeared in a sentence.

Pattern-matching (QGENR). Given a sentence in the form of a list, QGENR tries to generate a question from it and an expected answer. QGENR consists of three parts: (1) a preprocessor, (2) a pattern-matcher, and (3) a post-processing filter.

Preprocessing Stage. If the first word of the sentence says "Goodbye," "Quit," etc., then return, "I have enjoyed working with you. Goodbye."

If the sentence contains a colon, reject it. (Sentences with colons were often found to contain subsentences or complex phrases which the pattern matchers could not

TABLE 1
Question-Generating Patterns

| No. | Sentence Pattern | Question | Answer |
|-----|---|--|--|
| A | $\left\{ \begin{array}{l} \text{Prep} + Z +, + X + \left\{ \begin{array}{l} \text{Verb} \\ \text{Aux} \end{array} + Y \\ \text{or} \\ X + \left\{ \begin{array}{l} \text{Verb} \\ \text{Aux} \end{array} + Y + \text{Subord} + Z \end{array} \right.$ | <u>Case 1: X = nil or You ∈ X and Tense = present</u> What should you do + Subord + Z? | $\left\{ \begin{array}{l} \text{Verb} + Y \\ \text{Aux} \end{array} \right.$ |
| | | <u>Case 2: Verb is intransitive</u> What $\left\{ \begin{array}{l} \text{happens} \\ \text{happened} \end{array} \right.$ with + X + Subord + Z? | Verb + Y |
| | | <u>Case 3: First word of Y ≠ Verb, Aux, or "not"</u> What QForm (X + Verb) + Subord + Z? | Y |
| | | <u>Case 4: None of the above</u> What $\left\{ \begin{array}{l} \text{happens} \\ \text{happened} \end{array} \right.$ + Subord + Z? | Y |
| B | X + as + Y + as + Z | How + Y + QForm (X)? | Z |
| C | X + as + Y | How ~ QForm (X)? | Z |
| D | X + So That + Y | Why + QForm (X)? | Y |
| E | X + (Be) Used to + Verb + Y | What + (Be) + X + Used for? | to + Verb + Y |
| F | X + (Be) + NP + *Y | What + QForm (X + (Be))? | NP + *Y |
| G | X + To + Verb + Y | What + QForm (X) + to do? | Verb + Y |
| H | $Z + \text{Noun} + \left\{ \begin{array}{l} \text{Prep} + Y \\ \text{nil} \end{array} \right\} + \text{Relative} + *Y$ $+ \text{Verb}_2 + Z + \left\{ \begin{array}{l} \text{Verb}_1 \\ \text{Aux} \end{array} + *Y \right.$ | What + noun + $\left\{ \begin{array}{l} \text{Verb}_1 \\ \text{Aux} \end{array} + *Y \right.$ | $*Y + Y + *X + \text{Verb}_2$ + Z |
| I | X + Verb + Prep + Y | $\left[\begin{array}{l} \text{Where} \\ \text{When} \\ \text{What} \end{array} \right. + \text{QForm (X + Verb)} + \left[\begin{array}{l} \text{Nil} \\ \text{Nil} \\ \text{Prep} \end{array} \right. ?$ | Y |
| J | $*Y + \text{Noun} + \text{Prep} + Y + \left\{ \begin{array}{l} \text{Verb} \\ \text{Aux} \end{array} + Z \right.$ Where Prep ≠ of and Y contains no verbs | What + Noun + $\left\{ \begin{array}{l} \text{Verb} \\ \text{Aux} \end{array} + Z ? \right.$ | *Y + Y |
| K | $Y + \text{Noun} + \left\{ \begin{array}{l} \text{Verb} \\ \text{Aux} \end{array} + Z \right.$ Where Y ≠ Det and Y does not contain prepositions, coordinate conjunctions, or verbs | What kind of + noun + $\left\{ \begin{array}{l} \text{Verb} \\ \text{Aux} \end{array} + Z ? \right.$ | Y |
| L | X + That + Y | What + QForm (X)? | Y |
| M | Y + Verb + X and tense = past or present | <u>Case 1: Y contains pronouns</u> What + QForm (Y + Verb)? | X |
| | | <u>Case 2: Y contains no pronouns</u> What + Verb + X? | Y |

NOTATIONS: * in front of a variable indicates it is optional and may be omitted.
 $\left\{ \right.$ indicates alternative forms.
(Be) indicates any form of the verb "to be."
 $\left[\right.$ indicates corresponding alternatives
Aux = auxiliary = can, should, would, had, is, does, etc.
Subord = subordinate conjunction = although, because, besides, if, unless, whereas, while, until, when, whenever.
Det = determiner = a, an, the.
Prep = preposition.
QForm = a function which performs subject-verb inversion.

easily handle.)

If the sentence has more than 35 words, reject it. The figure "35" was arbitrarily chosen in order to eliminate the most complex and unwieldy sentences while permitting a fair number of sentences to remain for further processing.

If there are less than four words preceding a comma, which include a preposition or filler word like "nevertheless," "however," etc., strip them off the beginning of the sentence.

If the sentence begins with a comma or filler word, strip it off.

Mark all the words ending in -ly as adverbs.

Apply the function SETNOUN to every word in the sentence. This function first tries to see if the word is already in the vocabulary or is a verb form of a known verb. If so, it is marked as such; otherwise, it is marked as a noun.

If the first word in the sentence is a verb, the sentence is rejected.

The sentence is scanned for two verbs with the same tense and person connected by the word "and." If found, all the words are deleted starting with the first verb up to and including the "and."

The word preceding the verb is marked as a noun. Then the subroutine CHECKVERBS is executed to mark excess verb forms as nouns.

Pattern-matching Question Formation Stage. A list of patterns was developed by generalizing from the sentence features which I found myself using when I generated questions from text. Each pattern seemed to be associated with a particular rule for generating a question. The list of patterns is not claimed to be complete, optimal, or always correct.

The program checks the patterns given in Table 1 in sequence until a pattern is found which matches the sentence. Then the corresponding questions and answers are generated. Although the table does not show adverbs, they may optionally appear next to verbs in the sentence patterns. The program first checks to see if any subordinate conjunctions occur. If they do, only the first set of patterns in the table is handled. The question is generated from the main part of the sentence and the subordinate clause is appended to the end of the question.

Some comments on the patterns are in order. Pattern D really only works where the verb in Y is subjunctive. Pattern B could probably be eliminated, since G also generates a sensible question for infinitive

verb complements. F is an attempt to handle passive voice. H, J, K, and M 2 are all questions about the subject of the sentence which ask for the adjectives, prepositional phrases, or relative clauses which modify the principal noun in the subject noun phrase. Pattern I asks for the prepositional phrase which modifies the main verb (which must be passive or intransitive).

Pattern L works only when the "that" clause follows the main verb of the sentence and is the object of it, as in "the investigation showed that..." Pattern I asks for the object of the verb when the subject contains certain pronouns because to ask for the subject would be to invite a trivial answer.

Post-processing Stage. In order to screen out questions and answers which are likely to be of poor quality, several post-processing filters are applied to the generated question-answer pairs. The filters were somewhat arbitrary rules designed to eliminate the most complex questions and answers.

The program rejects the Q-A pair if any of the following conditions are true:

1. Either the question or answer is longer than 17 words.
2. The word "and" appears in the question.
3. The first word of the answer is a relative pronoun or subordinate conjunction.
4. Certain pronouns appear in the question.
5. Certain pronouns appear in the answer and the answer has less than five words.
6. There is a comma in the question.
7. An error was encountered previously in the subject-verb inversion routine QFORM.

Finding the Verbs in the Sentence (CHECK-VERBS). Possible verbs are separated from verbal forms used as nouns by the following process:

1. If a "verb" is immediately preceded by a determiner, it is marked as a noun.
2. If a "verb" immediately precedes another verb and the first verb is not a copula, it is marked as a noun.
3. A verb following an auxiliary, or an auxiliary plus an adverb, is retained.
4. A verb following the word "to" and possibly preceded by an adverb is retained.

5. A verb following some form of "be" or "have" and possibly preceded by an adverb is retained as a verb.

The number of remaining verbs is counted and compared with the number of relatives and subordinate conjunctions in the sentence plus one. If the difference is zero, the program returns the sentence as a success. If it is less than zero, the sentence is rejected as hopeless. If it is greater than zero, the remaining verbs are scanned from right to left. Present tense verbs which could be adjectives or nouns are marked as nouns until the number of excess verbs is reduced to zero if possible. If necessary, past tense forms are then marked as nouns until the number of excess verbs is reduced to zero.

Subject-verb Inversion (QFORM). QFORM performs subject-verb inversion according to the following rules:

1. If there are any conjunctions in the sentence, take the QFORM of the first part of the sentence and append the part with conjunctions.

2. If there is an auxiliary in the sentence, move it to the front of the sentence, and return. Look for a third person singular present tense verb. Replace it by infinitive forms and put "does" at the beginning of the sentence and return.

3. Look for a present tense verb and, if found, put the word "do" at the beginning of the sentence and return.

4. Look for a past tense verb and, if found, put the word "did" at the beginning of the sentence and return.

5. If none of the above work, set the global variable QFORMERROR, which signals failure to the system.

Paragraph Recognition

The input-output routines for AUTOQUEST turn out to be extremely complex and require nearly 50% of the processing time. The difficulty lies in recognizing exactly what a paragraph is. Usually the first word of a paragraph is indented, but sometimes it is not. If a paragraph is indented, this fact must be distinguished from the indentations of every line which are associated with margins. Some paragraphs are identified by letters or numbers preceding the first word. On the other hand, we do not want to treat a table of contents as if it were a paragraph. Some paragraphs are recognizable by an extra blank line between paragraphs, but the recognition algorithm must allow for double- or multiple-line spacing. Section headings or titles should not be treated as sentences. Copyright notices and similar paraphanelia are not good candidates for instruction and need to be screened out automatically.

Various heuristics have been devised to screen out noise from the text which should be processed. For example, a table of contents can usually be recognized by embedded blanks or periods in the middle of the line. Titles are usually eliminated by screening out "sentences" of less than five words.

The program stores each paragraph it reads in two forms: (1) as a list of sentences and (2) as a string of characters. Before generating a question, the program presents the paragraph to the student as a string, exactly as it was read. Then a sentence is randomly selected from the paragraph list. If it has already been used for a question, it may be rejected and a new sentence selected. The QGENR routine is executed to produce a question-answer pair.

RESULTS

The AUTOQUEST program was tested on a set of abstracts of technical reports from the Stanford University Artificial Intelligence Laboratory and on a page of text from an IBM programmer's manual. An example of a study session using AUTOQUEST appears in the appendix and illustrates both successful and unsuccessful interactions.

Of the 50 generated questions, 34 were judged to be satisfactory. The 16 unsatisfactory questions were classified into four groups: (1) eleven syntactic errors, (2) three semantic errors, (3) one pedagogical error, and (4) and one program bug. The syntactic and semantic errors are listed in full in Tables 2 and 3. The program bug was one in which the value DET was replaced by NOUN on the property list for the word "the," causing the program to generate a question in which the answer was the single word "the." The one error classified as pedagogical was the question, "What do we describe?," which could have been answered by two different sentences in the same paragraph.

TABLE 2
Examples of Syntactic Errors with Autoquest

| | |
|-----|---|
| 1. | SENTENCE: There are two usual characterizations of the least fixed-point of a continuous function. |
| | QUESTION: What are two usual characterizations of the least fixed-point of a continuous function? |
| 2. | SENTENCE: We present two different first order axiomizations of the metamathematics of the logic which FOL itself checks and show several proofs using each one. |
| | QUESTION: What kind of proofs using each one? |
| 3. | SENTENCE: "On Automating the Construction of Programs" 65 pages, May 1974. |
| | QUESTION: What May 1974? |
| 4. | SENTENCE: Methods are induced to detect some types of unmatchable target areas in the original data and for detecting when a supposed match is invalid. |
| | QUESTION: What happens with methods when a supposed match is invalid? |
| 5. | SENTENCE: Input to the memory has the form of analyzed conceptual dependency graphs which represent the underlying meaning of language utterances. |
| | QUESTION: What input meaning of language utterances? |
| 6. | SENTENCE: Then it looks at these lines and eliminates lines which cannot match any of the model lines. |
| | QUESTION: What cannot it eliminates lines which match? |
| 7. | SENTENCE: The method, embodied in a computer program, generates a complete list of isomers. |
| | QUESTION: What embodied generates a complete list of isomers? |
| 8. | SENTENCE: The program has the ability to create, destroy, and even resurrect objects in its world. |
| | QUESTION: What does the even resurrect object in? |
| 9. | SENTENCE: This progress report covers the first year and one half of work by our automatic programming research group at the Stanford Artificial Intelligence Laboratory. |
| | QUESTION: What work programming research group at the Stanford Artificial Intelligence Laboratory? |
| 10. | SENTENCE: The user retains control at all times. |
| | QUESTION: Where does the user retains control? |
| 11. | SENTENCE: COPILOT uses the multiple processing facilities to its advantage to achieve a kind of interactive control which we have termed < non-preemptive >. |
| | QUESTION: What does COPILOT uses the multiple process facilities to its advantage to do? |

TABLE 3
Examples of Semantic Errors with AUTOQUEST

| | |
|----|---|
| 1. | SENTENCE: The primary goal of comprehension is always to find meanings as soon as possible. |
| | QUESTION: How soon is the primary goal of comprehension always to find meanings? |
| 2. | SENTENCE: Although FACT uses substantially more main memory than MACRO-10, it assembles typical programs about five times faster. |
| | QUESTION: What does it assemble although FACT uses substantially more main memory than MACRO-10? |
| 3. | SENTENCE: The program is reproduced in full. |
| | QUESTION: What is the program reproduced in? |

The semantic errors seem to involve generating questions which imply a concrete answer where the source sentences involve abstract or idiomatic answers. Perhaps a simple addition of semantic markers (abstract-concrete, human-nonhuman, etc.) would prevent most such errors.

Nearly 69% of the errors were syntactic. By far the most common error was misidentification of the verb in the sentence. The source sentences are not themselves syntactically ambiguous. The problem seems to be that the pattern-matcher throws away most of the syntactic information in the sentence which a complete parser would utilize.

A second type of syntactic error, illustrated by sentences 4 and 6 in Table 2, is to mistake the level of a subordinate clause. For example, the when clause in sentence 4 is assumed to modify "included," but actually modifies "detecting." The pattern-matching approach is insensitive to levels of sentence embedding.

CONCLUSIONS AND RECOMMENDATIONS

The pattern-matching system works best on relatively simple sentences. It assumes that the verb which it finds is the main verb of the sentence, not part of some embedded sentence. The relative clauses, prepositional phrases, infinitive verb complements, and subordinate clauses which it is asking for must modify constituents of the top level of the sentence and not embedded sentences.

The system would work much better if a reliable parser could be found to serve as the front-end of the pattern matcher. It would not be necessary to parse the sentences down to the lowest level. All that is required is that the parser correctly identify the top level constituents of the sentence. A satisfactory parser for our purposes must have the following properties:

1. It must have complete error recovery capabilities.
2. It must be able to identify the top level constituents of the sentence.
3. It must be able to flag cases of multiple parsings if they occur at the top level.
4. It must be able to determine the grammatical category of unknown words from their usage.

The most important requirement is error recovery. It is not necessary for the parser to work all the time, provided we know when it is not working.

Since questions need to be generated only once or twice per paragraph, a great many sentences can be rejected if the parser encounters an error or ambiguity in them. About the worst thing a parser can do is generate a system diagnostic which forces a break.

The next worst thing it can do is pretend it has not encountered an error.

There is some reason to hope that a reliable parser meeting the above specifications will be available in about 2 years, in which case some very simple modifications to the AUTOQUEST system will improve its performance. For example, Kaplan and Kay have been working on a general syntactic processor which handles syntactic ambiguity in an economical format (Kaplan, 1973), and Burton (personal communication) has done some preliminary work on a grammar compiler which runs the NASA grammar about ten times faster than Woods' original system and which has improved debugging facilities (Burton, 1975).

For the near future, these parsers will have to operate with grammars that are capable of describing only 30 to 50% of English text sentences. Very little effort or funding is being expended on developing more complete computerized grammars of English.

The results with the present version of the system are sufficiently encouraging, however, so that a test of the pedagogical effectiveness of AUTOQUEST could and should be carried out in the future.

REFERENCES

- Alessi, S. M., Anderson, R. C., Anderson T. H., Biddle, W. B., Dalgaard, B. R., Paden, D. W., Smock, H. R., Surber, J. R., & Wietecha, E. J. Development and implementation of the computer-assisted instruction study management system (CAISMS). San Diego, California: Navy Personnel Research and Development Center, Technical Report TR 74-29, February 1974.
- Anderson, T. H., Anderson, R. C., Alessi, S. M., Dalgaard, B. R., Paden, D. W., Biddle, W. B., Surber, J. R., & Smock, H. R. A multifaceted computer-based course management system. San Diego, California: Navy Personnel Research and Development Center, Technical Report TR 75-30, April 1975.
- Anderson, T. H., Anderson, R. C., Dalgaard, B. R., Paden, D. W., Biddle, W. R., Surber, J. R., & Alessi, S. R. An experimental evaluation of a computer-assisted instruction study management system (CAISMS). San Diego, California: Navy Personnel Research and Development Center, Technical Report TR 75-31, April 1975.
- Anderson, R. C., & Biddle, W. B. On asking people questions about what they are reading. In G. Bower (Ed.), Psychology of Learning and Motivation, 9, 1975.
- Anderson, R. C., Surber, J. R., Biddle, W. B., Zych, P. M., and Lieberman, C. E. Retention of text information as a function of the nature, timing, and number of quizzes. San Diego, California: Navy Personnel Research and Development Center, Technical Report TR 74-28, February 1974.

Burton, R. R. Personal communication on 16 June 1975 at Bolt, Beranek, and Newman, Inc., Cambridge, Massachusetts.

Colby, K. M., Parkinson, R. C., & Frought, B. Pattern-matching rules for the recognition of natural language dialogue expressions. American Journal of Computational Linguistics, 1, M5, 1974.

Horn, G. K. P., & Winston, P. H. Personal computers. Datamation, 21, 111-115, 1975.

Kaplan, R. M. A general syntactic processor. Natural Language Processing, edited by R. Rustin, New York: Academic Press, 1973.

Schank, R. C., and Colby, K. M. Computer models of thought and language. San Francisco: W. H. Freeman & Co., 1973.

Simmons, R. F., & Smith, M. K. Generating and answering literal English questions from text. Austin, Texas: Department of Computer Science, Technical Report NL-21, University of Texas, March 1974.

Walker, D. E. Automated language processing. Annual Review of Information Science and Technology, 8, 69-119, 1973.

Walker, G. M. Video-disk battle goes public. Electronics, 48, 72-73, 1975.

Weizenbaum, J. ELIZA - A computer program for the study of natural language communication between man and machine. Communications of the Association for Computing Machinery, 9, 36-45, 1966.

Wilks, Y. An artificial intelligence approach to machine translation. Computer Models of Thought and Language, edited by Schank & Colby, San Francisco: W. H. Freeman & Co., 1973.

Winograd, T. Understanding natural language. New York: Academic Press, 1972.

Woods, W. A., Kaplan, R. M., & Nash-Webber, B. The lunar sciences natural language information system, Springfield, Virginia: National Technical Information Service, BBN Report #2378, June 1972.

APPENDIX SAMPLE AUTOQUEST DIALOGUE

*AIM-211, CS-383, AD769673,
Yorick Wilks,
"Natural Language Inference"
24 pages, September 1973.

The paper describes the way in which a Preference Semantics system for natural language analysis and generation tackles a difficult class of anaphoric inference problems (finding the correct referent for an English pronoun in context): those requiring either analytic (conceptual) knowledge of a complex sort, or requiring weak inductive knowledge of the course of events in the real world. The method employed converts

all available knowledge to a canonical template form and endeavors to create chains of non-deductive inferences from the unknowns to the possible referents. Its method of selecting among possible chains of inferences is consistent with the overall principle of "semantic preference" used to set up the original meaning representation, of which these anaphoric inference procedures are a manipulation.

ARE YOU READY TO ANSWER A QUESTION ON WHAT YOU HAVE JUST READ? Sure

WHAT DOES THE METHOD EMPLOYED ENDEAVOR TO DO?
Create chains of non-deductive inferences from the unknowns to the possible referents.

CORRECT

AIM-212, CS-384, AD769379,
Annette Herskovits,
"The Generation of French from a Semantic Representation"
20 pages, September 1973.

The report contains first a brief description of Preference Semantics, a system of representation and analysis of the meaning structure of natural language. The analysis algorithm which transforms phrases into semantic items called templates has been considered in detail elsewhere, so this report concentrates on the second phase of analysis, which binds templates together into a higher level semantic block corresponding to an English paragraph, and which, in operation, interlocks with the French generation procedure. During this phase, the semantic relations between templates are extracted, pronouns are referred and those word disambiguations are done that require the context of a whole paragraph. These tasks require items called <paraplates> which are attached to keywords such as prepositions, subjunctions, and relative pronouns. The system chooses the representation which maximizes a carefully defined "semantic density."

ARE YOU READY TO ANSWER A QUESTION ON WHAT YOU HAVE JUST READ? OK

WHAT CHOOSES THE REPRESENTATION WHICH MAXIMIZES A CAREFULLY DEFINED SEMANTIC DENSITY?

The author of the report.

WRONG -- SYSTEM / TRY AGAIN

The report contains first a brief description of Preference Semantics, a system of representation and analysis of the meaning structure of natural language. The analysis algorithm which transforms phrases into semantic items called templates has been considered in detail elsewhere, so this report concentrates on the second phase of analysis, which binds templates together into a higher level semantic block corresponding to an English paragraph, and which, in operation, interlocks with the French generation procedure. During this phase, the semantic relations between templates are extracted,

pronouns are referred and those word disambiguations are done that require the context of a whole paragraph. These tasks require items called <paraplates> which are pronouns. The system chooses the representation which maximizes a carefully defined "semantic density."

ARE YOU READY TO ANSWER A QUESTION ON WHAT YOU HAVE JUST READ? yes

WHAT CHOOSES THE REPRESENTATION WHICH MAXIMIZES A CAREFULLY DEFINED SEMANTIC DENSITY?
The system.
CORRECT.

A system for the generation of French sentences is described, based on the generation of French sentences is described, based on recursive evaluation of procedural generation patterns called <stereotypes>. The stereotypes are semantically context sensitive, are attached to each sense of English words and keywords and are carried into the representation by the analysis procedure. The representation of the meaning of words, and the versatility of the stereotype format, allow for fine meaning distinctions to appear in the French, and for the construction of French differing radically from the English origin.

AIM-213, CS-385,
Ravindra B. Thosar,
"Recognition of Continuous Speech: Segmentation and Classification Using Signature Table Adaptation"
37 pages, September 1973.

This report explores the possibility of using a set of features for segmentation and recognition of continuous speech. The features are not necessarily <distinctive> or minimal, in the sense that they do not divide the phonemes into mutually exclusive subsets, and can have high redundancy. This concept of feature can thus avoid apriori binding between phoneme categories to be recognized and the set of features defined in a particular system.

ARE YOU READY TO ANSWER A QUESTION ON WHAT YOU HAVE JUST READ? yes

WHAT CAN THE FEATURES HAVE?
high redundancy.
CORRECT

An adaptive technique is used to find the probability of the presence of a feature. Each feature is treated independently of other features. An unknown utterance is thus represented by a feature graph with associated probabilities. It is hoped that such a representation would be valuable for a hypothesize-test paradigm as opposed to a one which operates on a linear symbolic input.

ARE YOU READY TO ANSWER A QUESTION ON WHAT YOU HAVE JUST READ? yes

WHAT IS EACH FEATURE TREATED INDEPENDENTLY OF?
other features.
CORRECT.