Computational Linguistics

A. G. OETTINGER, Editor

Answering English Questions by Computer: A Survey

R. F. SIMMONS
System Development Corporation, Santa Monica, California

Fifteen experimental English language question-answering systems which are programmed and operating are described and reviewed. The systems range from a conversation machine to programs which make sentences about pictures and systems which translate from English into logical calculi. Systems are classified as list-structured data-based, graphic data-based, text-based and inferential. Principles and methods of operations are detailed and discussed.

It is concluded that the data-base question-answerer has passed from *initial* research into the *early* developmental phase. The most difficult and important research questions for the advancement of general-purpose language processors are seen to be concerned with measuring meaning, dealing with ambiguities, translating into formal languages and searching large tree structures.

I. Introduction

The last decade has seen many varied approaches toward computer processing of natural languages. The largest number of projects have been concerned with mechanical translation between languages. Document retrieval systems based on computers have become fairly common and more recently programs which support stylistic and content analysis of documents have been built. In the course of this still early development of natural-language-processing techniques, more than a dozen systems which attempt to answer English questions have been reported.

The term *question-answering machine* is used here rather loosely to include general-purpose language processors which deal with natural English statements and/or questions. These vary from conversation machines to machines which generate sentences in response to pictures, and systems which translate from English into logical calculi. All of these may be interpreted in some sense as attempting to use natural English in a manner very closely related to the question and answer pattern.

This survey was conducted by System Development Corporation's language processing research project, Synthex, under ARPA Contract SD-97.

Research toward natural language question-answering systems has a history dating only from 1959. Currently, 15 or 16 programs exist which answer some type of English question. It must be emphasized that all of these are experimental devices, often significant and exciting in their implications for future developments, but not in themselves practical devices to do the world's work. In a sense, the status of these question-answering systems is like the status of television in the 1930's; a number of breadboard devices exist, each of which shows that some aspects of the panorama of verbal meaning can be successfully reproduced by machine, but none as yet offers general solutions to the problem of high-quality language processing or attacks the engineering problems which a practical device would encounter.

A harshly critical review could say of each question-answering system so far built that it deals only trivially with a trivial subset of English. Nevertheless, each does answer some subset of English questions, and in the early stages of a research discipline, the effectiveness and generality of the systems developed is of considerably less interest than are the principles which emerge from the experimentation. In reviewing five years' accumulation of question-answering systems, this paper takes a tolerant viewpoint. It is concerned primarily with explaining and extracting principles and techniques of question answering and with communicating to a wider audience the state of the art of language processing.

II. A Logic of Questions and Answers

Most languages have a small set of rules which can be used for transforming any statement into a question. In English the question mark, intonation, and the rearrangement of subject and verb accomplish this function. In addition to transformational rules, a vocabulary of special question words—who, why, where, etc.—exists which provides clues as to the nature of the answer that is desired. Linguistic differences between declarative statements and questions are well catalogued and understood (Lees, 1960).

For logical differences the situation is not so clear-cut. A recent attractive idea is that a question is a special subclass of assertions whose propositional truth or falsity can be determined. Harrah (1961) and Belnap (1963) use this viewpoint as a basis for beginning a logical analysis of questions and answers. For opposing arguments see Hamblin (1963) and MacKay (1961). This logic offers both a consistent way of looking at questions and a classification scheme for questions and answers.

In the Belnap classification a question has two parts; one part delineates a *set of alternatives*, the other makes a *request*. The request part of the question indicates the acceptable form of a *direct answer* by showing which and how many of the alternatives must be present. A direct answer is that particular set of alternatives which are a complete answer to the question. For example, in the question, "What are two primes between 1 and 10?" the set of numbers between 1 and 10 are presented as an alternative from which an answer is to be selected. The request states that two and *any two* of these will be a direct answer. There exist also partial answers, eliminative answers, corrective answers and relevant answers. Each of these responses offers something less than the questioner hoped for.

Questions may also be classified as complete (disjunctive), e.g., "Is Brown the governor of California?" and as incomplete, "Who is the governor of California?" It is also desirable on occasion to classify questions as safe, risky, foolish, etc. A safe question is one that divides the universe in two as in "Did she wear the red hat or not?" Whatever in fact occurred there is a direct answer to this class of question. A foolish question is one which cannot have a direct answer, e.g., "What is the largest number?" Risky questions include those with built-in assumptions such as "Have you stopped beating your wife?"

Closely related to questions are imperative statements. "Go to the store," "Set course 180, speed 500," and "Set Dodgers equal win over Boston at Detroit" are all imperatives. Most often the imperative calls for a physical action not involving language while a question usually dictates a language response. In terms of question-answering programs however, the difference becomes mainly one of output mode; the analysis phase for each type of statement is similar. Like the question, the imperative contains a request and outlines an environment of alternatives. In such commands as "Name the signers of the Constitution," where the desired behavior is linguistic, the difference disappears entirely. An important implication for language-processing machines is that the logic developed for question-answering systems applies almost directly to machines for doing useful nonlinguistic work in response to English commands.

In summary, a question may be considered as a special subset of imperative statements where the desired response is linguistic. Questions are composed of two parts, one which describes the set of alternatives which include the answer and another which makes a request for a particular subset of these alternatives. A direct answer is a complete answer to the question and all other answers provide less information than the questioner desired. In the analysis of question-answering systems which follows, the main features of the Belnap classification system are either implicitly or explicitly recognized and dealt with. It will be seen that the distinction between commands and questions tends to be of minor importance for these machines.

III. Precursors

Machines and programs which attempt to answer English questions have existed for only about five years. But the desire to translate language statements into symbols which can be used in a calculus has existed as long as formal logic. Attempts to build machines to test logical consistency date back at least to Ramon Lull in the thirteenth century. Several logic machines for testing the validity of propositions were constructed in the nineteenth century. For the interested reader, Gardner's book *Logic Machines and Diagrams* (1958) offers a fascinating technical history of this line of development. However these machines, although they answer questions, do not deal directly with natural languages. Only in recent years have attempts been made to translate mechanically from English into logical formalisms and these will be briefly outlined in Section VII.

In this section two programs reported in 1959 will be described as foreshadowing the principles developed more fully in later question answerers.

The Conversation Machine. This program by L. Green, E. Berkeley and C. Gotlieb (1959) allows a computer to carry on a seemingly intelligent conversation about the weather. The problem was originally posed in the context of Turing's definition that a computer could be said to be thinking if it could carry on a conversation with a person in another room in such a manner that the person could not tell if he was speaking to a real person or not. By choosing a conversational topic as stereotyped as weather, the experimenters hoped to gain some experience with the meaning of Turing's idea.

The conversation machine dealt with three factors: meaning, environment, and experience. Meaning is expressed in terms of dictionary entries for words, combinations of these entries for remarks, and preference ratings (like or dislike) for certain types of weather. The environment of the system is an input of the day's weather, and its experience is a general knowledge of the type of weather experienced at various times of the year.

Words are categorized as *ordinary*, e.g., snow, rain, etc.; *time*, e.g., today, December, etc.; and *operator*, e.g., not, change, stop. The meaning of each word is stored as an attribute-value pair. For time words the attribute is type of time, calendar or relative, and the value is a code for the amount. For operator words the attribute is a code for a function to be accomplished and the value is the degree to which it is to be executed. Thus "change" and "stop" call a subroutine for negation but the degree code for "stop" is greater than that for "change." Such ordinary words as

"dew," "drizzle" and "rain" are coded for successively higher values of the attribute "wetness."

The meaning of a remark is calculated by looking up each word and coding it by its attribute-value pair from the dictionary. In the case of words not in the dictionary, defined as meaningless words, the code zero-zero is assigned. The set of codes for words in a remark represents its meaning. The program compares the meaning of a remark with its own store of knowledge and experience which has been similarly coded, then selects a stereotyped reply frame and fills in the blanks with words originating from the remark, from its experiences or from its preference codes.

As an example the authors present, "I do not enjoy rain during July." The "not" operator word acts on "enjoy" to give "dislike" which is a meaningful word to the program. The resultant meaning for the remark is the set, "dislike," "rain" and "July." Looking up the time word "July," the program discovers that "July" is associated with "heat" and "blue skies." Since these two terms do not relate to "rain," the program records an *essential disagreement*. On this basis it selects a reply frame and fills in the blanks as follows:

Well, we don't usually have *rainy* weather in *July*
so you will probably not be disappointed.

The conversation machine avoids the whole problem of syntactic analysis and is obviously limited to a few simple constructions. However it does manage to analyze a statement into a set of meaningful parameters which are then used to select an answer. Its principle of coding the meaning of words as an attribute-value pair is still basic to far more recent and advanced question systems.

The Oracle. As a Master's thesis under John McCarthy, then at M.I.T., A. V. Phillips programmed an experimental system to answer questions from simple English sentences (1960). Its mode of operation is to produce a syntactic analysis of both the question and of a corpus of text which may contain an answer. This analysis transforms both the question and the sentence into a canonical form which shows the subject, the verb, the object, and nouns of place and time. The system was written in LISP which simplified the programming task.

Its principle of operation can be appreciated by following the example in Figure 1. The example sentence is analyzed into subject, verb and (essentially) object. The analysis is limited to simple sentences and breaks down if the sentence has two or more subjects or objects. The first stage of analysis is to look up each word in a small dictionary to discover its word class assignment. At this point such words as school, park, morning, etc., are also coded as time or place nouns. During the analysis the question is transformed into declarative order and auxiliary verbs are combined with their head verbs so that both question and potential answering statement are in the canonical form, subject-verb-object, as shown in Figure 1.[1]

A comparison is then made to determine if the elements of the sentence match those of the question. In the example all three elements match and the program would print out "to school" followed by the entire sentence. Had the input been a complete question, i.e., "Did the teacher go to school?" the Oracle would have modified its behavior to respond "Yes."

As an early question answerer, the Oracle is a competent example of the principle of answering questions by structural matching of syntactic-semantic codes. Within the range of very simple English structures the method is uncomplicated and easily achievable. The principle of double coding—for syntactic and semantic word class—will be seen to generalize to much more complicated structures than Oracle used.

The conversation machine and the Oracle are two prototypes of question-answerers, which even in 1959 demonstrated that *if* statements could be coded semantically and syntactically they could be matched to discover how closely they resembled each other. For the conversation machine the match was against a coded data base and the selection of a reply to a remark was a function of the type of correspondence between the remark after coding and the program's coded knowledge. For the Oracle the comparison was between an English question and an English sentence, both of which were inputs.

IV. List-Structured Data-Base Systems

A *list-structured data-base question-answering system* is one that deals with data that are strongly organized into list form. SAD SAM reads Basic English[2] sentences and extracts from them data which can be appended to a list-structured data base. The Baseball system answers questions from lists which summarize a Major League's season's experience. The recent DEACON system is designed both to build a data structure from English sentences and to

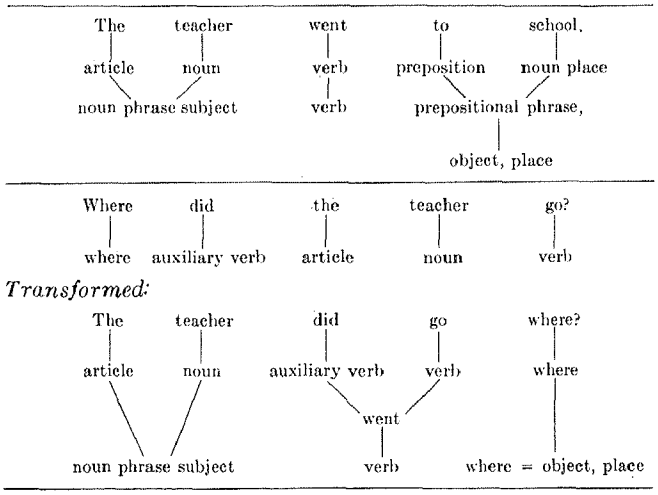

FIG. 1. Oracle-type analysis of question and answer

[1] Details of the types of syntactic analysis commonly used will usually not be discussed here. For a survey of the various methods used on computers, see D. Bobrow (1963).

[2] Basic English is a subset of English limited to a vocabulary of 300-1600 words and the simplest grammatical constructions. See Ogden (1962).

answer English questions from it. All of these programs illustrate and explore the principle of well-organized but limited information structures as a basis for experimenting with methods of answering English questions.

SAD SAM. This acronym stands for Sentence Appraiser and Diagrammer and Semantic Analyzing Machine. It was programmed in IPL-V by R. Lindsay (1963) as part of a dissertation at Carnegie Institute of Technology. SAD SAM is divided into two parts, a parsing section and a section for handling meanings. The system is designed to accept simple sentences limited to a Basic English vocabulary concerning family relationships. The data base is in the form of a family tree represented in the program by a hierarchical set of lists. As a sentence is read, it is parsed and the information that a person bears a relationship of brother, mother, father, etc., to someone else is extracted, and the name so represented is appended to the appropriate lists or branches of the family tree.

The parsing system is an independent program which uses a form of the predictive-analysis techniques which have been described in detail by Oettinger and Kuno (1963). Although it was designed for relatively simple structures, Lindsay reports that it can handle relative clauses and at least some appositional strings. As a result of the parsing, the input to the semantic analysis program is (1) a sentence whose parts are labelled noun, verb, noun phrase, etc., and (2) a tree structure showing the relationships among these grammatical features.

The semantic analyzer searches for subject-complement combinations which are connected by the verb "to be" and cross-references these to indicate that each is equivalent to the other. Words which modify such equivalent words are then grouped together. The vocabulary of Basic English provides only eight words to characterize kinship relations so these are then sought in the sentence. Thus, for the sentence,

John's father, Bill, is Mary's father.

the term, "John's father" would be set equivalent to the complement, "Mary's father." The two kinship terms would be recognized and the proper names which modify them would then be discovered. The word "Bill" modifies the subject and since subject and object are equivalent it also modifies the object. Triplets are constructed to show the relationship between each pair of names as follows:

Bill (father) John
Bill (father) Mary

These relationships are added to the family tree which then has the following structure:

Family unit (name)

(Attribute)	*(Value)*
Husband	Bill
Wife	Unknown
Offspring	John, Mary
Husband's parents	Unknown
Wife's parents	Unknown

Since this data structure is in the form of IPL lists, instead of actual names of family members, pointers may be used to indicate the location of a different family list which contains the names. The result is an interlocking data structure which allows a fairly significant level of inference. In the example above, since John and Mary are offspring of a common parent, it is known that they are siblings. If a following sentence states that Jane is Bill's wife, it will be immediately known that Jane is the mother of John and Mary (since no multiple marriages are permitted).

Lindsay's primary interest was in machine comprehension of English and he attempted to show that an important component of understanding lay in building large coordinated data structures from the text which was read. He found it necessary to use a syntactic analysis to discover relationships between the words which his program was able to understand and then to transform the portions of the sentence which were understood into a form which could map onto his data structure.

Baseball. This is a program originally conceived by Frick, Selfridge and Dineen and constructed by Green, Wolf, Chomsky and Laughery (1963). It answers English questions about the scores, teams, locations and dates of baseball games. The input questions are restricted to single clauses without logical connectives such as "and," "or" or "but" and excluding such relation words as "most" or "highest." Within the limitations of its data and its syntactic capability, Baseball is the most sophisticated and successful of the first generation of experiments with question-answering machines. It is of particular interest for the depth and detail of its analysis of questions.

Baseball is programmed in IPL and uses list structures to organize data. The data are set up with a major heading of months. For each month there is a list of places in which games were played. For each place there is a list of days, for each day a list of games, and for each game a list of teams and score values, exemplified by the following data format:

Month = July
  $Place_1$ = Boston
    Day_1 = 7
      Game Serial # = 96
        Team = Red Sox, Score = 5
        Team = Yankees, Score = 3

The program also contains a dictionary which includes the part of speech of a word, its meaning, an indication of whether it belongs to an idiom, and a code to show if it is a question word. The first part of the program's task is to use the dictionary, parsing routines and content routines to translate from the English language question into a specification (or spec) list which is similar in format to the data structure.

The first step is to substitute dictionary codes for the English words. A parsing using a modification of Zellig Harris's approach (1962) results first in bracketing the phrases of the question, then in determination of subject, object and verb. For example, the question "How many

games did the Yankees play in July?" gives the following bracketing:

(How many games) did (the Yankees) play (in (July))?

The brackets distinguish noun phrases and prepositional phrases and locate the data which are needed for the spec list. The parsing phase resolves some ambiguities of the noun-verb type while others such as "Boston = place" or "Boston = team" are resolved later. Some, of course, are not resolvable.

A semantic analysis phase actually builds the spec list from the parsed question. In this phase the dictionary meanings of the words are used. The meaning may be an attribute which is part of the data structure, as in "team" means "team = (blank)," or "who" means "team" = ?"; or the meaning may be a call to a subroutine, as for example "winning" means "routine A1" which attaches the additional condition "winning" to "team" on the spec list. The output of these routines is a spec list which is used to search the list structures of the data store for an acceptable answer.

After the spec list is completed, the processing phase takes over. In some cases, this requires the simple matching of a blank item on the spec list such as the place in which a given team played on a given day. In other cases, as with the words "every," "either," and "how many," processing is a very complicated searching and counting procedure. The output of the program is in the form of a *found* list which shows all of the acceptable answers to the question.

In the Baseball system three aspects of the question-answering problem stand out clearly. A first phase of syntactic analysis merges into the second phase, semantic analysis. However, for the first time a third logical processing phase becomes explicit. In this phase, even though the relations between words and the meaning of words are already known, a wide range of operations are performed as a function of these meanings.

Having considered the manner in which Lindsay's SAD SAM reads text to append data to a list structure similar to that used by the Baseball system, it is apparent that Baseball could become a completely self-contained (though limited) automatic language processor. To achieve this goal, factual statements would be read and analyzed into their spec lists and a new processor would be required to add the data to the storage lists.

The DEACON Breadboard. At General Electric's TEMPO, F. Thompson (1964) and J. Craig (1963) have reported on DEACON[3]—a data-based question answerer which is part of a man-machine communication system that may eventually allow operators to communicate with each other and with the computer in a subset of natural English. The question answerer is programmed in a special list-processing language, KLS-II, developed at TEMPO for this system. At this writing, many aspects of the natural-language programs have been checked out and the authors expect a complete question answerer to be operable soon.

In general, DEACON depends on a list-structured data base. Thompson makes explicit the importance of a well-understood data structure and introduces a principle of equivalence between the word classes of syntactic analysis and the semantic categories of the data base. As a result his programs do not break neatly into a parsing system, a semantic analyzer, and a data processor, although these phases are still distinguishable. His language analysis parses a sentence into the names of lists and the calls to operations to be performed on the lists. These operations are performed immediately and the resulting sublists are tested for their truth value in the last phase of data processing.

An example presented by Thompson (1964, pp. 17–23) will clarify the operation of these programs.

Question: The cost of what Air Force shipments to Omaha exceeds 100,000 dollars?

The data base from which this question is to be answered is outlined in Table 1. This table can be read as a list structure of shipments with sublists of Air Force, Army and Navy shipments. Each of these sublists has attributes of cost, origin and destination and values as shown in the cells of the table.

The first step in analyzing the question is that of assigning word classes as follows:

the	*F*	to	*F*
cost	*A*	Omaha	V
of	*F*	exceeds	R(V, V)
what	*F*	100,000	N
Air Force	*M*	dollars	A
shipments	*L*	?	

The word "shipment" is classified L as a major list. The term "Air Force" is classified M as a list modifier (or major sublist). "Cost" and "dollars" are assigned A for attribute. "Omaha" is designated V for value and "100,000" is designated N for number. Each of the words coded F represents some function for the system to perform. The parsing is accomplished with a phrase structure grammar in which each rule is accompanied by a transform to operations on the data list structure. For example, the following rule,

$$L_1 = M + L : T_1(M,L)$$

TABLE 1

Shipments	*Cost*	*Origin*	*Destination*
Air Force			
Shipment a	$ 39,000	Detroit	Washington
Shipment b	103,000	Boston	Omaha
Army			
Shipment a	—	—	—
Shipment b	—	—	—
Navy			
Shipment a	—	—	—
Shipment b	—	—	—

[3] DEACON stands for Direct English Access and CONtrol.

means that when the combination of word classes $M + L$ is found, substitute for L_1 the list which is generated by T_1 operating to extract the sublist M from the major list L. The word "what" is a function word interpreted as a quantifier which generates all cases of the structure which it modifies. At the conclusion of the analysis there is the rule,

$$T = V+R(V,V) + V : T_7(V,R,V).$$

This rule translates roughly into "Is it true that the value of the data generated in the first clause exceeds the value of the data generated in the second clause?"[4] By the time this rule is applied, the first and second clauses are each represented by a list. $R(V,V)$ is a function which tests a member of the first list as greater in value than a member of the second. In this case the second list has one entry, value 100,000 dollars, and for each member of the first list the indicator for true or false is assigned as a result of the comparison.

The TEMPO system accepts the occurrence of ambiguous analyses but usually these are resolved in terms of the data context of the sentence. Each remaining analysis is dealt with as a separate statement or question. It generalizes to a broad range of data and to a reasonably complex subset of English. The system is self-contained in that it both reads its own data and answers questions. It makes explicit the principles of structure and question analysis which although previously implicit in such systems as Baseball, SAD SAM and the PLM were not then fully conceptualized. It is theoretically important in showing the continuity between syntactic, semantic and symbolic-logical analyses of English in a data base system.

Other Data-Base Approaches. Work by Walker and Bartlett (1962), and by Sable (1962) and several classified systems under development by the Air Force are further examples of attempted systems which query a data base in some more or less restricted set of English. The outline of a special problem-oriented language for translating from a subset of English into operations on a data base was reported by Cheatham and Warshall (1962) and other computer language developments are also of interest in this area. In general, however, these are not primarily studies of question answering in natural language.

V. Graphic Data-Base Systems

Two interesting systems which depend on graphic data bases are described in this section. One clearly shows the power of translating from English or from graphic data into a subset of the predicate calculus. The other takes a probabilistic learning approach toward the generation of valid English statements from the diagrams it reads. Together they add further support to the idea, suggested earlier by Lindsay (1963) and others, that a well-structured data base offers great potential for making inferences as well as for providing explicitly stored data.

The Picture Language Machine (PLM). At the National Bureau of Standards, R. Kirsch (1964), D. Cohen (1962), B. Rankin (1961) and W. Sillars (1963) have devised a program system which accepts pictures and sentences as input. It translates both the pictures and the English statement into a common intermediate logical language and determines whether the statement about the picture is true. This set of programs is of particular interest in this review since it seems to be one of the first to explore the principle of translating from a subset of English into a formal language and to point out a reasonable method for doing so.

The PLM is composed of three subsystems—a parser, a formalizer, and a predicate evaluator. Its language is limited to a small subset of English suitable for making statements and asking questions about three geometric figures. The parsing system is based on an immediate-constituent grammar that includes the discontinuous-constituent operator. Parsing is accomplished by a recognition routine which successively substitutes symbols for the dictionary for words in the sentence or for an intermediate symbol string until the top of the parsing tree is reached.

After the sentence has been parsed to produce one or more tree structures representing it, the formalizer translates the parsed sentence into the formal language. The formal language is a first-order functional calculus with a small number of constant predicates. The primitives of the language include brackets, parentheses, the terms "and," "if . . . then," "for all," "there exists," "not," "identity," certain other quantifiers, variables and finally three types of predicates. The singular predicates are typified by the following examples:

Cir(a)	a is a circle
Bot(a)	a is at the bottom
Bk(a)	a is black

Some typical binary and ternary predicates follow:

Bgr(a,b)	a is bigger than b
Lf(a,b)	a is to the left of b
Smc(a, b)	a is the same color as b
Bet(a,b,c)	a is between b and c
Mort(a,b,c)	a is more to the right of b than c
Mmid(a,b,c)	a is more in the middle of b than c

The formalizer is designed to work with each parsing that the grammar produces. For each rule in the grammar there is a corresponding rule of formalization. The translation process is primarily one of substituting formalization symbols for grammar symbols beginning at the top of the parsed tree and working down. More than simple substitution is required to insert quantifiers and implication symbols, but essentially the process of translating to the formal language bears a great similarity to that of generating a language string from a phrase structure grammar. For each parsing of a sentence the translation into the

[4] For example, is clause 1 (cost of AF shipments, etc. = 103,000) greater than clause 2 (100,000 dollars = 100,000); thus, is 103,000 < 100,000?

formal language results in a unique, unambiguous, well-formed formula. An example sentence "All circles are black circles" has only one parsing which finally translates into the following formal statement:

$$(\forall X_1)\,[\mathrm{CIR}(X_1) \supset (\exists X_2) \cdot [\mathrm{CIR}(X_2)\ \&\ \mathrm{Bk}(X_2)\ \&\ (X_2 = X_1)]].$$

The structure of this formula is explicit and unambiguous; the relationships between the geometric variables are clearly specified and the truth value may be tested by the predicate evaluator. If true, the answer to the implied question "Are all circles black?" is *yes*.

The predicate evaluator translates from pictures to the formal language. It is designed to accept inputs that have been processed by SADIE, a scanning device which is used as an input to a computer. The inputs are limited to three sizes each of triangle, square or circle, each of which may be in outline or filled in. A technique called *blobbing* is used to distinguish objects resulting from the scan and each such object is then circumscribed with a rectangle. Maximum and minimum x-y coordinates are computed and the ratios of these serve to distinguish triangles from circles or squares. Circles are distinguished from squares on the basis of covering less area. A *black* figure is one whose area is filled in while a *white* figure is an outline. These relatively simple computations suffice to generate the valid predicates from the picture matrix.

Some of the interesting features of the PLM can be appreciated only by close study of its documentation. For example, the grammar used is a modified phrase structure system with a remarkably compacted notation (Kirsch, 1963). The problem of ambiguity of syntactic analysis is accepted and each possible interpretation of the sentence is tested for validity as a well-formed formalization. There is a practical scheme for translating at least a small subset of English into the predicate calculus and an equally feasible system for testing the formalization against that resulting from the picture matrix.

Namer. Simmons and Londe (1964) at SDC programmed a system to generate natural-language sentences from line drawings displayed on a matrix. The primary intent of this research was to demonstrate that pattern recognition programs could be used to identify displayed figures and to identify the relationships among them. After this had been established, a language generator was used to generate simple sentences such as "The square is above, to the right of, and larger than the circle." The sentences that are generated are answers to the various relational questions which could be explicitly asked.

The pattern recognition aspects of Namer were derived from work by Uhr and Vossler (1963). When a picture is presented on the input matrix, a set of 96 characteristics is computed. The algorithms or operators compute these as functions of the size, shape and location of the pattern in the matrix. Typical characteristics that are derived include one-bit indications of the presence or absence of parts of the figure in sections of the matrix, of protuberances, of holes in the pattern (as in a circle), and of indentations as in a "u." A first-level learning stage of Namer selects a small subset of the 96 characteristics—those which correlate most highly with correct recognition of the name by which the experimenter designates the pattern.

The second level of Namer operates in a comparable fashion to obtain characteristics of the sets of coordinates representing two patterns. At this level the operators generate characteristics of comparative size, separation, density, height, etc. Subsets of these 96 characteristics are learned in the same fashion as at the earlier level to correlate with such relation terms as above, below, thicker than, to the right of, etc.

The language generator uses a very brief phrase structure grammar to generate simple sentences which are true of the picture. For example,

> The dog is beside and to the right of the boy.
> The circle is above the boy.
> The boy is to the left of and taller than the dog.

There is a great variety of drawings that can be learned and once a relationship is learned between any two figures it usually generalizes successfully to most other pairs of figures.

Both the PLM and Namer show the capability of making geometric inferences based on a set of computational operations on line drawings. Unique sets of characteristics resulting from the computations can be mapped onto English names and words expressing spatial relationships. In this respect these systems anticipate some recently developed inference systems (see Section VII on SQA). The PLM has the additional feature of being able to answer questions about those English language statements which are permissible in its grammar. Since it can translate from English into a formal statement, the formalization for the question can be compared to that for the proposed answer. By the addition of predicates which go beyond simple spatial relations, the PLM may generalize into a much broader inference system than any yet available.

Namer on the other hand is not strictly a question answerer. In order to answer English questions selectively it would be necessary to match the valid statements that can be generated against an analysis of the specific question. However, Namer offers a probabilistic learning approach for learning names and relationships in a data base. This approach may generalize far beyond spatial relationships and their expression in language and may suggest a method for dealing with inference in nongraphic data bases as well.

VI. Text-Based Systems

In the previous two sections, question answerers which query a well-structured but limited data base have been described. The text-based systems, in contrast, attempt to find answers from ordinary English text. As a conse-

quence, neither the language to be used nor the data to be queried lend themselves to fractionation into convenient small packages of vocabulary or simple syntax. Although current experimental text-based systems in fact deal with relatively small amounts of text (from 100–500 thousand words) they are designed with much larger amounts in mind. To some extent they resemble more ordinary information retrieval systems in their use of indexing and term-matching techniques, but they deal at the level of English questions and sentences instead of term sets and documents. In addition, the text-based question answerer adds linguistic and semantic processing phases to evaluate the material discovered in the retrieval phase.

Three such systems will be described here—Protosynthex, the Automatic Language Analyzer and the General Inquirer. Since this area is closely related to that of fact and document retrieval systems, references will be provided to lead the interested reader deeper into that area.

Protosynthex. At SDC, Simmons and McConlogue with linguistic support from Klein (Simmons, Klein, McConlogue, 1963) have built a system which attempts to answer questions from an encyclopedia. The problem in this system was to accept natural English questions and search a large text to discover the most acceptable sentence, paragraph or article as an answer. Beginning at the level of ordinary text, Protosynthex makes an index, then uses a synonym dictionary, a complex intersection logic, and a simple information scoring function to select those sentences and paragraphs which most resemble the question. At this point, both the question and the retrieved text are parsed and compared. Retrieved statements whose structure or whose content words do not match those of the question are rejected. A final phase of analysis checks the semantic correspondence of words in the answer with words in the question.

Beginning with natural text that has been keypunched, an indexing pass is made and an index entry is constructed for each content word in the text. A root-form logic is used to combine entries for words with similar forms; for example, only one index entry exists for govern, governor, government, governing, etc. The contents of the entry are a set of VAPS numbers which indicate the Volume, Article, Paragraph and Sentence address of each occurrence of the indexed word.

The first step in answering the question is to look up all of its content words in the index and so retrieve all of the appropriate VAPS numbers. At this stage a dictionary of words of related meaning is used to expand the meaning of the question's words to any desired level. Thus the question, "What animals live longer than men?" might result in the following lists of content words as a query to the index.

Word	*Words of Related Meaning*
animals	mammals, reptiles, fish
live	age
longer	older, ancient
men	person, people, women

The intersection test finds the smallest unit of text, preferably a sentence, in which the greatest number of words intersect. A simple information score based on the inverse of the frequency of occurrence of the word in the large sample of text is used to weight some words more heavily than others in selecting potential answers. All of this computation is done with the VAPS numbers that were obtained from the index. The highest scoring five or ten potential answers are then retrieved from the tape on which the original text was stored. These comprise an information-rich set of text which roughly corresponds to the set of alternatives proposed by the question (within limits of the available text).

The question and the text are then parsed using a modification of the dependency logic developed by D. Hays (1962). For example, Figure 2 shows the dependency structures of a question and some potential answers which were retrieved. In passing it should be noted that the parsing system learns its own word classes as a result of being given correctly analyzed text. The human operator interacts frequently with this parser to help it avoid errors and ambiguities. In the simple examples in Figure 2, the principle of dependency structure matching can be seen. All of the potential answers included "worms" and "eat," the content words from the question. But only those statements in which the dependency of "eat" on "worms" is maintained need be considered further as possible answers.

The actual matching is accomplished by a fairly complex set of programs which build a matrix containing all the words from the question and from its possible answers. The

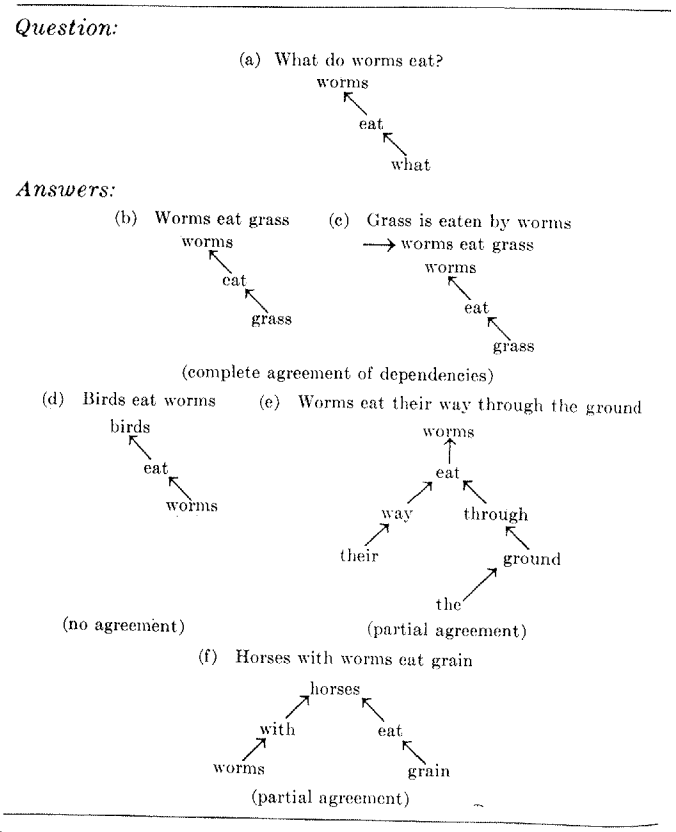

FIG. 2. Dependency structure of a question and some potential answers PROTOSYNTHESIS

matrix is examined to discover the structure matches and the part of the statement which corresponds to the question word. For the example question, this evaluation program would output as follows:

worms = worms
eat = eat
what = grass
what = their way
what = through the ground

A semantic evaluation system is now required to score each of the words in phrases corresponding to "what." This system is essentially a dictionary lookup whose entries can grow as a function of use. If certain words are found to be answers to "where" questions they will be so coded in the dictionary. If the question had been "Worms eat what food?" the words "ground," "way" and "grass" would have been looked up and compared in semantic coding with "food." Those which corresponded most closely would have been scored as best answers. The semantic evaluation system is still in early stages of experimentation but is expected to resemble the parsing system in that its dictionary will be developed and modified as a function of experience with text under the control of an online operator.

The approach of Protosynthex is to successively filter out more and more irrelevant information, leaving ultimately only statements which have a high probability of being answers to the question. This system is an attempt to deal with a large and syntactically complex sample of natural text. It is a symbiotic system in which a man works with the computer to help resolve both syntactic and semantic ambiguities. Only in this fashion is the program able to overcome the problems associated with multiple, apparently valid interpretations of the same sentence or question.

The Automatic Language Analyzer (*ALA*). From Indiana University a series of quarterly reports by Householder et al. (1960–62) and a final technical report by Thorne (1962) describe the progress toward completion of a rather complicated automatic language analysis system.[5] This system is designed to handle the breadth and complexity of language found in a book on astronomy. As a question-answering system, it introduces a variation of the principle of translating from English into an intermediate language which bears a strong relationship to dependency structure. When translated to the intermediate language, FLEX, the question or text is also augmented by semantic codes obtained from *Roget's Thesaurus*—or from a specially constructed thesaurus. The degree of matching between question and text is then computed to select a best answer.

The primary information store for the ALA is a preanalyzed set of sentences stored on tape. The preanalysis includes assignment of FLEX codes and of thesaurus references. The thesaurus is a list of clusters each of which indexes the portions of the text in which members of the cluster appear. A dictionary of word-stems and phrases provides cross-references to clusters in which the word appears. The sequence of operations is that the question is first analyzed and assigned FLEX and thesaurus codes, then sentences are selected and matched, and finally the paragraphs that contain supposed answering sentences are printed out with their scores.

The transformation of English into the FLEX language is begun by looking up each word in a dictionary to assign ordinary syntactic word classes. At this point a great deal of effort is spent to resolve word-class ambiguity by use of special routines which use additional cues available in the sentence. The next phase is to order the words into clauses and phrases and to check the accuracy of this ordering. The breaking into clauses is accomplished by the use of marker words such as verbs and absolute markers such as because, how, if, what, when, etc. When the sentence has been analyzed into subject, verb and their qualifiers, the translation into FLEX is accomplished as shown below.

The old man ate stale food reluctantly.

*S*1	*S*2	*P*1	*P*2	*P*3
man	old	ate	food	stale
			reluctantly	

The notation is to be read, "*S*1 means subject, *S*2 is the first qualifier, *P*1 means the verb, and *P*2. . . . *Pn* refer to verb modifiers." The importance to the sentence of each FLEX symbol is rated separately for subject and predicate in order of the numbers assigned. Thus an *S*1 or a *P*1 are most heavily weighted in the later comparison process.

Each word also carries a semantic coding. This code is simply a list of the thesaurus clusters in which it is found. For semantic matching of words a and b, the following formula is used:

$$\text{Semantic Correlation} = \frac{n_{ab}}{\sqrt{(n_a \times n_b)}}.$$

where n_a and n_b are respectively the number of clusters in which a and b are found, and n_{ab} is the number they share in common. When a question and a sentence are matched, *mutual relevance* is scored by considering the following three comparisons in a weighting scheme: (1) relative importance of the category (i.e., *S*1 or *P*1 more important than *S*2 *P*2), (2) match of FLEX category (*S*1 = *S*2 better than *S*1 = *S*3, etc.), and (3) cluster matching score (according to the above formula). The paragraphs containing the best scoring sentences and their scores are then recovered and printed out.[6]

Although programming of this system is apparently not yet completed, and it may be claimed that the FLEX transformation leaves much to be desired as an intermediate language, the ALA is unquestionably one of the more ambitious and sophisticated systems so far described. At this stage of experimentation it is worth wondering how well the semantic correlations will in general correspond to meaning matches between statements. In any case it is a

[5] For the sake of convenience, this has been abbreviated to the ALA system and the present summary is based on the final report by Thorne.

[6] For other associative scoring techniques see (Doyle 1963).

clearly formulated realization of what has hitherto been a rather vague idea that a thesaurus may be helpful in question answering.

The General Inquirer. A paper by P. Stone et al. (1962) at Harvard University describes a COMIT program system useful for analyzing the content of text. As a question-answerer, the General Inquirer recovers all sentences containing a given set of concepts. As in the Householder-Thorne ALA, a thesaurus is used for coding words as to concept membership and, if desired, an intermediate language may be used which makes explicit the syntax of the text and question. However, the General Inquirer differs from most of the systems so far described in that any syntactic manipulations are done as a manual pre-editing phase for the text and the questions.

Probably the most interesting feature about these programs is the dictionary and thesaurus operation. The thesaurus is built especially for the content to be studied. For example, a thesaurus for psychological studies includes headings such as Person, Behavioral Process, Qualities, etc. As subheadings under Behavioral Process, such cluster tags as the following are found: react, see, hear, smell, defend, dream, escape, etc. For an anthropological study the thesaurus would contain many different headings.

The dictionary includes about 3000 common English words and words of special interest to any particular investigation. The dictionary lookup is accomplished by first filtering out function words such as and, or, to, of, etc., then looking up the remaining portion of text (about 50 percent) in the complete dictionary. Each of the words in the main dictionary is defined by the thesuarus tags or clusters to which it belongs. Thus the dictionary entry for "abandon" has the following format:

ABANDON = GO+REJECT+END+DANGER+ALONE

In the processing phase each word in the text to be examined may be tagged by its cluster memberships and matched against the terms of the request.

Content analyses may be as simple as frequency counts of tag-concepts in a discourse or they may be requests for all sentences in the discourse with the tags "reject" and "person." A great deal of useful analysis can be accomplished using just the semantic portion of the General Inquirer. However, to avoid apparent matches which are structurally dissimilar (as in the overworked example, man bites dog vs. dog bites man) a syntactic analysis is often desirable.

The following interesting semantic-syntactic categories are used in the manual pre-editing phase.

1. Subject and incorporated modifiers
2. Nonincorporated subject modifiers
3. Predicate verbs
4. Verb modifiers including time referents
5. Object and incorporated modifiers
6. Nonincorporated object modifiers
7. Indirect object and modifiers
8. Attributive nouns
9. Attributive verbs

The following example (from a suicide note) will illustrate how these codes are actually used.

IN THE LAST/4 WEEK/4 A NUMBER/1
OF OCCURRENCES/1 HAVE FORCED/3 ME/5
INTO A POSITION/4 + WHERE 1/8 FEEL/9
MY LIFE/1 IS NOT WORTH/3 CONTINUING/3.

The "+" separates two clauses which were coded as separate "thought sequences." For the case of pronouns or ellipses the referent words are added in parentheses. It is to be noticed that the grammar distinguishes between metaphrases such as "I feel that" and declarative statements about apparent facts.[7] Like the Householder-Thorne ALA, the General Inquirer finds that only a limited syntactic analysis is sufficient for its purpose.

Additional applications of the General Inquirer or simulations of it can be found in North et al. (1963), and Ford (1963). It has already proved itself to be a system useful in supporting several types of content analysis.

Remarks on Text-Based Systems. The three systems described in this section introduce some principles not usually dealt with in data-base machines. The first of these is a semantic principle of indexing large bodies of text at a depth such that words in a question or their semantic tags can serve as queries to the index. The text processors each faced the problem of bringing explicit organization into the relatively loose flow of ordinary English and each attempts to solve it by use of some combination of indexing, and syntactic and semantic analysis. The dependency structure matching principle of Protosynthex and the translation into the FLEX language of ALA or the "thought sequences" of the General Inquirer are each examples of analyses which go beyond the purely syntactic. In all three systems, additional semantic coding is explicitly undertaken. The data organization in each of these systems is distributed among indexes, thesauruses and synonym dictionaries, and among the rules for text analysis. Nevertheless a strong data organization is present although information categories are overlapping rather than unique as in the data base questioners.

Other Text Questioners. Swanson (1963) has attempted to measure the effectiveness of natural language queries for retrieving documents. G. Salton (1962) has considered aspects of the structure-matching principle and means for measuring its degree in a given pair of sentences. Guiliano (1962) and J. Spiegel et al. (1962) have given a great deal of consideration to the use of word associations as a basis for question-answering and retrieval systems. There is some indication that associative indexing may prove a valuable adjunct to other techniques of comparing questions and their answers. The association nets described by Doyle (1963) also offer an interesting point of view toward question answering. Finally the whole area of classification systems, autoabstracting and document retrieval can be expected to contribute increasingly toward the development of general-purpose question answerers.[8]

[7] With codes 8 and 9, *attributive* meaning metaphrase.

[8] See also a recent discussion by Jane Robinson (1964).

VII. Logical Inference Systems

An approach of demonstrated usefulness, transformation from English to a quasi-logical formulation is represented by one long-term project and by five of the most recently developed question-answering programs. The earliest work in this area is reported in a paper by T. Williams (1956) concerned with translating from natural English to the predicate calculus. More recent work by Williams (1962) and a series of papers by Bohnert (1962, 1963) develop algorithms for translating from important small subsets of English into predicate calculus forms. Bohnert's Project LOGOS at IBM (Yorktown Heights) has developed computer programs which do this translation for such samples of English as a simple language to be used in a marriage bureau and a "war novel" command language. Bohnert's 1963 paper is notable for the clarity with which it presents arguments for the desirability and feasibility of making such transformations as well as for its development of several algorithms for handling difficult aspects of English.

The five most recently developed question-answering systems resemble each other in their use of explicit rules of logical or mathematical inference. Four of these are associated with artificial intelligence or linguistic laboratories at M.I.T. All are programmed in forms of COMIT or LISP, and all suggest a line of descent from McCarthy's Advice Taker model (1959) and bear a relationship to the family of problem-solving and theorem-proving systems.

The Darlington Logic Programs. A series of COMIT programs by J. Darlington (1964) at the M.I.T. Mechanical Translation Laboratory translate a subset of English into various forms of the propositional and predicate calculi. They then test the validity of arguments using a modification of the Davis-Putnam algorithm. The propositional system is able to discover clause separations and so recognize propositions using only an elementary type of syntactic analysis. A more complete phrase structure analysis is accomplished in programs concerned with translating relational statements into the predicate calculus.

The first phase of the translation is a dictionary lookup which assigns one of two word classes to each word in the text. A word is either a P for punctuation or a W for all other types. The P type are such words as the following: if, then, and, or, but, therefore, either, neither, nor, that, and several others which have significance as logical separators. At later stages additional word class subscripts are assigned. The P words are further subcategorized as to the type of connection or separation that they perform and as to the particular subroutine which is to be called for special processing. For example, "or," "nor" and "and" call a special subroutine which looks for "either . . . or," "neither . . . nor" and "and . . . both," respectively. The W words are assigned syntactic word classes such as: noun plural, auxiliary verb, adjective, gerund, etc.

At this point the verbs and gerunds are simplified into abbreviated infinitive forms (to be = be), and some sentences are expanded. For example, "The box and the chair are wooden" is processed into the transform, "The box be wooden and the chair be wooden." Every string of W words separated by a P word is then examined to determine if it is a sentence. These strings qualify as sentences only if each contains at least one noun and one verb. Propositional symbols such as A/V, B/V, etc., are then substituted for each string of W words following a P word. These strings are then compared and, where identical, are assigned the same propositional symbol.

Each proposition is then parenthesized according to a fairly complex set of priority rules in accordance with the P words which separate them. The output is then ready to feed into the program which uses the modified Davis-Putnam algorithm to test the validity of the propositions. The modifications to this algorithm simplify and speed up the process of testing consistency by eliminating redundant clauses.

In the functional logic translation program, sentences are parsed into a phrase structure tree whose nodes are labelled in a manner that makes transformation to quasi-logical formulae a simple process. For example the sentence "All circles are figures" transforms into the quasi-logical formula,

All + Variable A + Noun Phrase $1A$ + is + some + Variable A + Noun Phrase $2A$.

This formula is then transformed to the logical format by looking up its elements in a table of equivalents. The logical analysis proceeds through three levels. At the most detailed level all elements of the sentence will have been transformed into relational terms and existence assertions. Darlington is presently working toward the development of an algorithm which the system can use to choose automatically the level of analysis required to prove or disprove the argument.

The Darlington system can be appreciated as a specialized question answerer which tests a verbal argument for internal consistency. The example "All circles are figures. Therefore all who draw circles draw figures." was translated into logical form and proved valid in a total of about .3 minutes. (Note that "Do all who draw circles draw figures?" is the question that is being answered.)

Weaknesses of the Darlington system are the usual ones of limited ability to handle such complexities of English as pronomial reference, ellipsis, metaphor, etc., and its use of a tiny subset of the language. Its approach to syntactic analysis tends to be rather rough-and-ready using assumptions which over-simplify the problem and cast some doubts on the generality of the solutions. However, the system does exemplify a well-rounded attack on the problem of translating a small segment of English into logical notation and then evaluating the arguments. It indicates quite clearly that logical translators are an interesting and profitable line of research toward the aim of high-quality language processing by computers.

The Cooper System. An elegant example of translation from simple English statements into Aristotelian logic has

been programmed in COMIT at IBM (San Jose, Calif.) by William Cooper (1964). This system accepts a small subset of statements such as "Magnesium is a metal," "Gasoline burns rapidly," and "Magnesium oxide is a white metallic oxide." Such statements, or questions in the same form, are translated into one of the four basic Aristotelian sentence types, "All x is y, no x is y, some x is y, and not all x is y." The resulting logical form is tested for deducibility from the information already in the system.

Cooper very carefully defines the subset of English and of English grammar with which he is working. This subset includes adjectives, modifying nouns, substantives, "is" predicates and intransitive verb predicates. He also defines a logical language L^*. Within this logical language, a certain amount of syllogistic inference is possible. The translation algorithm includes phases of syntactic analysis and transformation into forms belonging to L^*. In rough outline the process of parsing and transformation is comparable to those already described in the Darlington system and the Kirsch PLM (1964). The system correctly answers "false" to the assertion "sodium chloride is an element" having already been given the statements, "sodium chloride is salt," "salt is a compound," and "elements are not compounds." It also discovers that "magnesium is a metal that burns rapidly" is true, after finding in its information store that "magnesium is a metal" and "magnesium burns rapidly."

Cooper's system is a carefully thought out and well-described example of translating from a subset of English into Aristotelian syllogistic logic. Obvious limitations in the generality of the approach, the usefulness of the subset of English, and the efficiency of the proof algorithm do not detract from the propaedeutic value of a system simply devised and clearly executed.

In this area of translation from natural language to logical formalisms, books by Reichenbach (1947) and Copi (1959) are basic texts. Recent papers by Kochen (1962), Krulee et al. (1962) and Travis (1963) each discuss important aspects of the problem of building question-answering systems which are based on formal languages.

The Specific Question Answerer (SQA). At Bolt Beranek and Newman, F. Black (1964) programmed the SQA to find brief answers to short English questions. It is called a *Specific Question Answerer* because it can extract only brief specific answers that are either directly stated in or deducible from its corpus of text. The program consists of a basic system written in LISP and a corpus which contains formal and informal rules of inference as well as declarative statements. In most of his examples Black avoids the problems of syntactic analysis by requiring exact matching of structure between questions and possible answers. (In the latest version, the input is in a formal language.)

The corpus is organized as a set of conditional statements of the form, "If —— then ——." A declarative statement such as "Mercury is a planet" is considered to be a conditional with an empty antecedent. A typical corpus is made up of such declarative statements and such complete conditionals as the following:

Mercury is next smaller than Pluto
Pluto is next smaller than Mars
Mars is next smaller than Venus
If X is next smaller than Y, then X is smaller than Y.
If X is next smaller than Y and Y is smaller than Z then X is smaller than Z.

Primitives of the system include variables such as X, Y, etc.; words (which include variables as well as English words such as Mars, Venus, is, next, etc.); and phrases which are strings of words.

The system operates by comparing a question with each of the *consequents* in the corpus. For example, the question "What is next smaller than Pluto?" matches immediately with the first consequent of the corpus, "Mercury is next smaller than Pluto." If the question were "Pluto is smaller than what?," the answering process is more difficult. In this case there are two matches with the consequents, "X is smaller than Y" and "X is smaller than Z." The word "Pluto" may immediately be substituted for X in each consequent. The first antecedents for each of these two consequents are then used as questions, i.e., "If Pluto is next smaller than Y and Y is smaller than Z" for the first question and "If Pluto is next smaller than Y (then Pluto is smaller than Y)" for the second. (Parentheses indicate that the enclosure is not an antecedent.) It will be left as a tree-searching exercise for the reader to follow the resulting net to obtain the answers, "Mercury and Venus."

Recent work with minor modifications to this program has shown that it can solve at least some of the Advice Taker problems (McCarthy 1959). Black has also suggested that the parsing of English syntax may be dealt with by conditional substitutions. For example, one question transform is "If X is a Y, then is X a Y." This rule transforms from one form of question to one form of declarative statement. To what extent such a transformational approach will actually account for English syntax remains to be supported by experimental evidence. Although the inference approach is undeniably powerful, two drawbacks remain. The first is that the SQA requires an exhaustive search of the network of matching consequents, and with even a few hundred consequents the time requirement must be very large. The second is the apparent requirement that the corpus of consequents be internally consistent. However, Black has suggested approaches toward easing these difficulties.

The Semantic Information Retriever (SIR). From the point of view of exploring a model of meaning as communicated in natural language, B. Raphael (1964) has built a program which accepts a class of simple English statements and, in interaction with the questioner, answers some questions about them. The model considers English words as objects and certain relationships as holding between them. The formalization of the model is a limited relational calculus. The model also takes advantage of

the property list characteristics of LISP. Although only a few meanings are specifically dealt with, Raphael suggests that by modeling words in their interrelationships, the meaning is preserved for a human reader.

This program avoids the complexities of syntactic analysis by limiting itself to a small number of fixed formats for sentences. The present system recognizes about 20 simple sentence formats which include both interrogative and declarative types. By comparison with these basic patterns an input sentence such as "Every boy is a person" is translated into the logical form: SETR (Boy Person). This means that "person" is in a superset relation to "boy." If a sentence or question does not correspond to a known format, it is rejected by the program with an appropriate comment to the operator.

Figure 3 shows a set of example inputs to this program and the resulting data structure for them. In answering the question, "How many fingers are on John?" the system is able to match the question form "(finger, John)" successively with "(finger, hand)," "(hand, person, 2)," "(John, boy)" and "(boy, person)." The "How many" requirement was not fully satisfied so it asks for further information. With the additional data "(finger, hand, 5)" in the numerical part-whole relation, it is able to calculate the answer, 10.

The Raphael program is another example of a system which uses a limited set of logical predicates such as subset, part-whole, left-right, etc., to allow study of deducing or inferring answers to questions. Like the Black program this one essentially ignores syntactic problems, and depends on internally consistent data. However, Raphael's model tests a sentence for consistency before accepting it as data and it also makes explicit the interaction with the questioner.

Both the SQA and SIR are examples of deductive systems which understand some aspects of the meaning of words. They put particular emphasis on various relational terms and use rules of logical inference to follow trees of axioms and theorems. In both cases, if the statement form of the input question can be deduced from information in memory, the answer is "yes;" if its negation is deduced, the answer is "no." If the statement cannot be deduced the answer is "don't know." SIR recognizes when information is missing and requests it, but is limited to those relational terms for which it has corresponding functional routines.

Input Statements:	*Formalization:*
Every boy is a person	SETR (Boy, Person)
John is a boy	SETR (John, Boy)
Any person has two hands	PARTRN (Hand, Person, 2)
A finger is part of a hand	PARTR (Finger, Hand)
Question:	
How many fingers are on John?	PARTRNQ (Finger, John
Computer Response:	
How many fingers per hand?	
Input Statement:	
Every hand has five fingers	PARTRN (Finger, Hand, 5)
Answer:	
The answer is 10	

FIG. 3. An example from the Raphael program

The SQA seems to be a more general approach, in that it can accept a very broad range of relational terms without the necessity of reprogramming. That is, the SQA, following the Advice Taker paradigm, allows the question asker to program the machine by giving it additional information. Both systems provide relatively simple and comparatively efficient algorithms for deducing answers to questions.

Student. D. Bobrow (1964) for an M.I.T. doctoral thesis has programmed an algebra problem solver which accepts problems phrased in a limited subset of English and transforms these into equations which can be solved arithmetically. The limited subset of English is mainly sufficient to account for the phrasing found in a high-school algebra text. The system is programmed in LISP and is currently operable on the timeshared 7094 computer system at M.I.T.'s Project MAC.

Student is based on a theoretical relational model whose objects are variables such as words, numbers, or phrases which name numbers. The relations are the ordinary arithmetic operations of adding, subtracting, multiplying, dividing, exponentiation, equality, etc. The means for expressing the relations among objects are sets of simultaneous equations. The problem that Student attacks is that of transforming a set of English statements in which a set of equations is implicit into an explicit formulation of those equations.

Background data which help Student to "understand" the meaning of certain words and phrases are provided by a part of the system which accepts simple English statements such as "twice always means two times" or "three feet equals one yard." This subprogram builds what is essentially a dictionary of transformations from one form of English into an equivalent formalism which the program can use, or into a form which is identical with a form used previously in a problem.

Bobrow's first step in transforming an English statement into a set of equations is to make mandatory substitutions such as "two times" for "twice," "square" for "the square of," and several others. His next step is to identify terms such as "plus," "percent," "times," etc., and to tag them as operators. In addition to tagging operators in this phase, the program also identifies certain verbs, question words, and the terminal question mark. The process is accomplished by dictionary lookup. After these operations an example problem appears as follows:

```
IF THE NUMBER (OF/OP) CUSTOMERS TOM (GETS/VERB)
  IS 2 (TIMES/OP 1) THE (SQUARE/OP 1) 20 (PERCENT/OP
  2) (OF/OP) THE NUMBER (OF/OP) ADVERTISEMENTS
  (HE/PRO) RUNS, AND THE NUMBER (OF/OP)
  ADVERTISEMENTS (HE/PRO) RUNS IS 45,
  (WHAT/QWORD) IS THE NUMBER (OF/OP)
  CUSTOMERS TOMX (GETS/VERB) (QMARK/DLM)
```

A most critical phase of the processing is the next step, in which two simple heuristics are used for breaking the problem statement into simple sentences. The first is to look for an "if" followed by anything followed by a comma followed by a question word and transform it to two sentences. For example:

If ... customers Tom gets ... is 45, What is ... QMARK

transforms to

... customers Tom gets ... is 45. What is QMARK.

The second heuristic, applied after the first, is to divide strings followed by ", and" into two simple sentences. After these operations the following simple sentences result from the example above:

THE NUMBER OF CUSTOMERS TOM GETS IS 2 TIMES THE SQUARE

20 PERCENT OF THE NUMBER OF ADVERTISEMENTS HE RUNS.

THE NUMBER OF ADVERTISEMENTS HE RUNS IS 45.

WHAT IS THE NUMBER OF CUSTOMERS TOM GETS QMARK

The operators and function terms are underlined in the above sentences.

All of the simple sentences are now of the form, "*P*1 is *P*2." In the first sentence, *P*1 represents "The number of customers Tom gets" and *P*2 is the remainder following "is." (In cases of sentences in the form "*X* does/has *Y*" the program transforms them to "The *thing X* does/has *is Y*.")

Each of the operators now calls for a special function to be performed. For example (OF/OP) checks to see if "of" is immediately preceded by a number; in that case it will be treated as the multiplication operator. Otherwise its operator tag will be stripped off and it will be ignored. A (PERCENT/OP2) looks at the preceding number and divides it by 100. Other such operators are not only more complicated but require consideration of precedence levels. The result of applying these operations is to put the equations into the explicit form that LISP can use in solving them.

Many hazards may still exist to block the explicitness of the equations. Because of pronouns, ellipsis, shift in measuring units (as from feet to inches, or from dollars to cents), or because of synonymic reference such as "people for customers," the variables and units in the equations may not match. If these difficulties arise, Student takes recourse to its memory of background data which can be manipulated by the user to provide unit transformations, sentence transforms, and synonyms. Heuristics for guessing the referent of pronouns are also provided.

There are numerous interesting features about Student, not the least of which are its use of a fund of background information and its substitution of a heuristic approach to sentence analysis for the more usual analytic one. The meaning of a problem is first resolved into simple sentence units, then into variables and operators. The meaning of the operators is the function they cause to be performed. For variables, the only pertinent question is "does the string *Pi* identically correspond to the string *Pj*." If not, possible transformations are considered to bring about such a result.

Although the system is obviously limited to a well-controlled, specialized subset of English, it has proved sufficiently versatile to solve a large number of high school algebra word problems. It contributes significantly to language-processing technology in its heuristic approach to syntactic analysis, its use of background information, and its direct approach toward translating a small class of English operator words into their mathematical equivalents.

VIII. Analysis and Conclusions

Principles of Question-Answering Systems. Although the question answerers described express a wide variety of approaches, there are in fact striking similarities in processing requirements. A strong organization of data storage is a common requirement. (Even in the case of English text questioners an additional phase of index processing is usual to obtain this degree of organization.) The language must be analyzed syntactically and in the more sophisticated systems semantic analysis is also required. Question and data are transformed into some canonical form and a matching and sometimes a scoring is undertaken to determine whether or not an answer is present. In the list-structured data-base systems, the output is usually a list of the matching terms from the lists which were the referent of the question. In addition, for each of the systems, there may be a level of inference-drawing although only a few make this level explicit.

Data Structures. The data-base programs understand text and/or questions by transforming them into the categories and subcategories of a well-structured data storage system. Usually this data system is in terms of lists in which each headlist has sublists which have attributes and values. The chief advantage of this structure is that a given item may be a member of many lists which address it (and so interrelate it) while the item need be recorded only once. The papers by Lindsay (1963) and by Thompson (1963) make explicit the value and importance of coding information by the structure in which it is embedded. The resulting structural relations permit some degree of inference.

In the text-processing programs, the natural English language text has an additional organization imposed upon it through a preprocessing phase of indexing, usually supported by lists of synonyms and a dictionary of concept codes. The data structure in these systems is less explicit and less centralized but quite as essential as in the list-structured data-base programs.

Syntactic Analysis. Although this paper has not emphasized the description of the special techniques of syntactic analysis, it is an important phase in all question-answering systems. (Even in those where simplifying assumptions are

used to avoid it!) Generally an immediate-constituent model is used and generally the problem of ambiguous interpretations is encountered. In the data-base systems, where a small subset of English constitutes the vocabulary, the problems of ambiguity sometimes can be resolved in terms of the referent data structures. In the text-processing systems, great effort is taken to resolve syntactic ambiguity before attempting to answer the question. Whether either of these approaches to resolving problems of ambiguity will in fact prove successful remains to be shown.

Semantic Analysis. The phase of a question-answering program devoted to semantic analysis is often not clear-cut and usually merges into the syntactic analysis and matching phases. In the data-base systems the meaning of a word is usually the denotation of either a data category or of a subroutine. This meaning is typically a coded entry associated with the word in a dictionary. In the text-processing systems the meaning of a word is generally a list of attribute codes (as the Roget cluster numbers in the Householder-Thorne ALA). By controlling syntactic correspondence the correlation or commonalities of meaning for corresponding words in two statements can be measured in terms of corresponding codes and accumulated across the sentences.

While semantic analysis is much more clearly understandable in terms of the data-base systems, it is apparent that not all subtleties of language can map onto an unambiguously defined data structure. The matter of coding the meanings of words and calculating the meanings of sentences is an area in which only the dimmest of understanding currently exists. However, theoretical work by Katz and Fodor (1963) and by Quillian (1963) suggest avenues of approach, and surely a direct attack on problems of meaning is probably most profitable.

Obtaining and Evaluating Answers. Syntactic and semantic analyses are undertaken to transform the question and the answering text into some canonical form in which they can be easily compared. Generally this form is a syntactically-ordered list of the semantic units with which the system deals. In the data-base systems, the standard form is a set of list names; in the text processors the units may be English words, their synonyms or semantic codes corresponding to words or terms. Several of the more recent systems use something approaching the format of the predicate calculus as a standard form.

However, even when both question and answer exist is canonical form, there are still serious problems in comparing the two. In the data-base systems there is generally the problem of actually processing the data structure and in some cases making inferences (such as, if X and Y are "offspring" of the same parent they are siblings—or, is 300,000 more than 100,000?). In the text-processing systems really good comparisons of questions and possible answers depend on far better semantic coding than has been presently developed and on a much-increased understanding of how to order the units for comparison. Although these systems are also potentially adept at making language inferences (by generating and answering additional questions and by using dictionaries of rules of linguistic inference) actual experiments in this direction are still largely lacking.

In Kirsch's PLM and Thompson's TEMPO systems, the meaning of a statement is accumulated substructure by substructure and the result is finally tested for truth value. In Student, the Algebra Word Problem Solver, the portions of meaning which are pertinent to solving the algebra problem are accumulated until a well-formed mathematical formulation is available. These systems certainly suggest the way in which meaning (as far as a particular machine function is concerned) can be extracted and accumulated. However to generalize these suggestions to a fairly large set of English words and constructions remains a formidable task for the future.

Outlook. Although research on question-answering systems is only five years old, it is possible to conclude that already enough important principles are understood to offer assurance that the next five or ten years will be even more rewarding. For immediate development the large data-base system, controlled and queried by a small subset of English which can vary freely, looks most promising. Until now data-base systems have worked with a very limited variety of data. The time seems ripe to experiment with systems which have a wide variety of data in their structure. Such systems would unavoidably be rather large, but all indications of data-base research to this time indicate that their problems can be mastered.

The outlook for text-processing systems is also promising but it would be overoptimistic to expect any developmental model with practical utility for some time to come. Questions of semantic coding, of accumulating and representing the meaning of statements, and of performing inference effectively are still very much in the initial research stage. The two general systems which so far exist (Protosynthex and the ALA) are intriguing early demonstrations of ultimately valuable language processors. Those systems which study question-answering methods by using rules of inference are a most recent and potentially attractive feature of the language processor, but a great deal of research into rules of logico-linguistic inference and into methods of translating from English into clarifying forms such as the predicate calculus remains to be done.

In summary, steady, even rapid, progress is being made toward the development of practical question answerers and general language processors. The most difficult questions are now becoming apparent. How does one characterize the meaning of a sentence? How are ambiguous interpretations, both syntactic and semantic, to be dealt with? How are inferences to be made without exhaustive tree searches? How are partial answers, widely separated in the text, to be combined? To what extent can we or should we translate from English into formal languages? Can these studies be attacked from a theoretical point of

view or do they yield best to the empirical approach of building large question answerers and language processors as test vehicles? Even partial answers to these questions will contribute to the eventual development of high quality, general-purpose language processors.

*

Acknowledgments. There is today an "invisible college" of language-processing researchers. Those engaged in research on question answering are a relatively new and close-knit group in this college. Thanks to the intensive exchange of information among members of this group, this review of question-answering systems shows fewer of my own biases and gaps of knowledge than would otherwise be true.

I am particularly indebted to Russell Kirsch who called my attention to several approaches to question answering which might not otherwise have been cited. Larry Travis has attempted to allay some of my ignorance in the area of translating from English into formal languages. Dan Bobrow (1963) by his competent survey, *Syntactic Analysis of English by Computer*, has saved me a great deal of detailed exposition of how each question answerer did its parsing. Discussions with several of the researchers cited in this review have helped me understand their various approaches.

RECEIVED MAY, 1964; REVISED OCTOBER, 1964

REFERENCES

BELNAP, N. D., JR. An analysis of questions: preliminary report. Doc. TM-1287, System Development Corp., Santa Monica, Calif., June 1963.

BLACK, F. S. A deductive question-answering system. Ph.D. Thesis, Div. Eng. Appl. Phys., Harvard U., Cambridge, Mass., June 1964.

BOBROW, D. G. Natural language input for a computer problem-solving system. Proc., Fall Joint Comput. Conf. *25* (1964). Also available as Ph.D. Thesis, Math. Dept., MIT, Cambridge, Mass., 1964.

——. Syntactic analysis of English by computer—a survey. Proc. Fall Joint Comput. Conf. *24* (1963), Spartan Books, Baltimore, 365–387.

BOHNERT, H. G. An English-like extension of an applied predicate calculus. AFOSR-TN-62-3, IBM Corp., Yorktown Hts, N. Y., Feb. 1962.

——. Logical-linguistic studies for machine text perusal. Semi-Ann. Proj. LOGOS, IBM Corp., Yorktown Hts, N. Y., Tech. Status Rep., May-Dec. 1963.

CHEATHAM, T. E., JR., AND WARSHALL, S. Translation of retrieval requests couched in a "semiformal" English-like language. *Comm. ACM 5*, 1 (Jan. 1962), 34–39.

COHEN, D. A recognition algorithm for a grammar model. Rep. 7885, Nat. Bur. Stand., Washington, D. C., Apr. 1962.

——. Picture processing in a picture language machine. Rep. 7885, Nat. Bur. Stand., Washington, D. C., Apr. 1962.

COHEN, F. What is a question? *Monist 39* (1929), 350–364.

COOPER, W. S. Fact retrieval and deductive question-answering information retrieval systems. *J. ACM 11*, 2 (Apr. 1964), 117–137.

COPI, I. *Symbolic Logic.* Macmillan, New York, 1959.

CRAIG, J. A. Grammatical aspects of a system for natural man-machine communication. RM63TMP-31, TEMPO, General Electric Co., Santa Barbara, Calif., July 1963.

DARLINGTON, J. L. A COMIT program for the Davis-Putnam algorithm. Res. Lab. Electron., Mech. Transl. Grp., MIT, Cambridge, Mass., May 1962.

——. Translating ordinary language into symbolic logic. MAC-M-149, Proj. MAC Memo., MIT, Cambridge, Mass., Mar. 1964.

DOYLE, L. B. The microstatistics of text. *J. Inform. Storage and Retrieval 1*, 4 (1963), 189–214.

FORD, J. D., JR. Automatic detection of psychological dimensions in psychotherapy. Doc. SP-1220, System Development Corp., Santa Monica, Calif., July 1963.

GARDNER, M. *Logic machines and diagrams.* McGraw-Hill, New York, 1958.

GIULIANO, V. E. Studies for the design of English command and control language system. Rep. CACL-1, Arthur D. Little, Inc., Cambridge, Mass., June 1962.

GREEN, B. F., JR., WOLF, A. K., CHOMSKY, C., AND LAUGHERY, K. Baseball: an automatic question answerer. In *Computers and Thought*, E. A. Figenbaum and J. Feldman (Eds.), McGraw-Hill, New York, 1963, 207–216.

GREEN, L. E. S., BERKELEY, E. C., AND GOTLIEB, C. Conversation with a computer. *Computers and Automation 8*, 10 (1959), 9–11.

HAMBLIN, C. L. Questions. *Australian J. Phil. 36*, 3 (1958), 160–168.

——. Questions aren't statements. *Phil. Science 30* (1963), 62–63.

HARRAH, D. A logic of questions and answers. *Phil. Sci. 28*, 1 (1961), 40–46.

HARRIS, Z. S. *String analysis of sentence structure.* Mouton, The Hague, Netherlands, 1962.

HAYS, D. G. Automatic language data processing. In *Computer Applications in the Behavioral Sciences*, H. Borko (Ed.), Prentice-Hall, Englewood Cliffs, N. J., 1962, 394–423.

HOUSEHOLDER, F. W., JR., LYONS, J., AND THORNE, J. P. Quart. rep. on automatic language analysis, 1–7. ASTIS, Indiana U., Bloomington, Ind., 1960–1962.

KATZ, J. J., AND FODOR, J. A. The structure of a semantic theory, part I. *Lang. 39*, 2 (1963), 170–210.

KIRSCH, R. A. The application of automata theory to problems in information retrieval (with selected bibliography). Rep. 7882, Nat. Bur. Stand., Washington, D. C., Mar. 1963.

——. Computer interpretation of English text and picture patterns. *Trans. IEEE-EC* (Aug. 1964). In press.

——, RAY, L. C., CAHN, L., AND URBAN, G. H. Experiments in processing pictorial information with a digital computer. Rep. 5713, Nat. Bur. Stand., Washington, D. C., Dec. 1957.

——, AND RANKIN, B. K., III. Modified simple phrase structure grammars for grammatical induction. Rep. 7890, Nat. Bur. Stand., Washington, D. C., May 1963.

KOCHEN, M. Adaptive mechanisms in digital "concept" processing. Proc. Amer. Inst. Electr. Eng., Joint Autom. Contr. Conf., 1962, 50–59.

KRULEE, G. K., KUCK, D. J., LANDI, D. M., AND MANELSKI, D. M. Natural language inputs for a problem-solving system. *Behav. Sci. 9*, 3 (1964), 281–288.

KUNO, S., AND OETTINGER, A. G. Syntactic structure and ambiguity in English. Proc. Fall Joint Comput. Conf. *24* (1963), Spartan Books, Baltimore, 397–418.

LEES, R. B. The grammar of English nominalizations, Part II. *Intntl. J. Amer. Ling., 26*, 3 (1960).

LINDSAY, R. K. Inferential memory as the basis of machines which understand natural language. In *Computers and Thought*, E. A. Feigenbaum and J. Feldman (Eds.), McGraw-Hill, New York, 1963, 217–233.

MACKAY, D. M. The informational analysis of questions and commands. In *Information Theory, Fourth London Symposium*, Colin Cherry (Ed.), Butterworths, Washington, D. C., 1961, 469–477.

McCarthy, J. Programs with common sense. Proc. Symp. Mechanisation of Thought Processes, I, London, England, HM Stationary Off., 1959, 75–91.

North, R. C., et al. A system of automated content analysis of documents. Rep. of Stanford Studies in International Conflict and Integration, Stanford U., Stanford, Calif., Mar. 1963.

Ogden, C. K. *The General Basic English Dictionary*. W. W. Norton, New York, 1962.

Phillips, A. V. A question-answering routine. Memo. 16, Artif. Intell. Proj., MIT, Cambridge, Mass., May 1960.

Quillian, R. A notation for representing conceptual information: an application to semantics and mechanical English paraphrasing. Doc. SP-1395, System Development Corp., Santa Monica, Calif., Oct. 1963.

Rankin, B. K., III. A programmable grammar for a fragment of English for use in an information retrieval system. Rep. 7352, Nat. Bur. Stand., Washington, D. C., June 1961.

Raphael, Bertram. SIR: a computer program for semantic information retrieval. Fall Joint Comput. Conf. *25* (1964). Also available as Ph.D. Thesis, MIT, Math. Dept., Cambridge, Mass., June 1964.

Reichenbach, H. *Elements of symbolic logic*. MacMillan, New York, 1947.

Robinson, J. J. Automatic parsing and fact retrieval: a comment on grammar, paraphrase and meaning. Memo. RM-4005-PR, RAND Corp., Santa Monica, Calif., Feb. 1964.

Sable, J. D. Use of semantic structure in information systems. *Comm. ACM 5*, 1 (Jan. 1962), 40–42.

Salton, G. Manipulation of trees in information retrieval. *Comm. ACM 5*, 2 (Feb. 1962), 103–114.

Sillars, W. An algorithm for representing English sentences in a formal language. Rep. 7884, Nat. Bur. Stand., Washington, D. C., Apr. 1963.

Simmons, R. F. Synthetic language behavior. *Data Process. Management, 5*, 12 (1963), 11–18.

——, and Londe, D. Namer: a pattern recognition system for generating sentences about relations between line drawings. Doc. TM-1798, System Development Corp., Santa Monica, Calif., Mar. 1964.

——, Klein S., and McConlogue, K. L. Indexing and dependency logic for answering English questions. *Amer. Documentation 15*, 3, (1964), 196–204.

——, and McConlogue, K. L. Maximum-depth indexing for computer retrieval of English language data. *Amer. Documentation, 14*, 1, (1963), 68–73. Also available as Doc. SP-775, System Development Corp., Santa Monica, Calif.

Spiegel, J., Bennett, E., Haines, E., Vicksell, R., and Baker, J. Statistical association procedures for message content analysis. In *Information System Language Studies, Number 1*, SR-79, MITRE Corp., Bedford, Mass., Oct. 1962.

Stone, P. J., Bayles, R. F., Namerwirth, J. Z., and Ogilvie, D. M. The general inquirer: a computer system for content analysis and retrieval based on the sentence as a unit of information. *Behav. Sci., 7*, 4 (1962), 1–15.

Swanson, D. R. Interrogating a computer in natural language. In Proc. IFIP Cong., Munich, 1962, C. M. Popplewell (Ed.), North Holland, Amsterdam, 1963, 288–293.

Thompson, F. B. Semantic counterpart of formal grammars. TEMPO General Electric Co., Santa Barbara, Calif.

——, et al. DEACON breadboard summary. RM64TMP-9, TEMPO General Electric Co., Santa Barbara, Calif., Mar. 1964.

Thorne, J. P. Automatic language analysis. ASTIA 297381, Final Tech. Rep., Arlington, Va., 1962.

Travis, L. E. Analytic information retrieval. In *Natural Language and the Computer*, P. Garvin (Ed.), McGraw-Hill, New York, 1963, 310-353.

Uhr, L., and Vossler, C. A pattern recognition program that generates, evaluates, and adjusts its own operators. In *Computers and Thought*, E. A. Feigenbaum and J. Feldman (Eds.), McGraw-Hill, New York, 1963, 251–268.

Walker, D. E., and Bartlett, J. M. The structure of laguages for man and computer: problems in formalization. Proc. First Cong. Inform. System Sciences, Sess. 10, MITRE Corp., Bedford, Mass., Nov. 1962.

Williams, T. M. Translating from ordinary discourse into formal logic: a preliminary study. ASTIA 98813, Avion Div., ACF Indust., Alexandria, Va., Sept. 1956.

——, Barnes, R. F., and Kuipers, J. W. Discussion of major features of a restricted logistic grammar for topic representation. Lab. Rep. 5206-26, ITEK, Lexington, Mass., Feb. 1962.

COMMENTS

By Vincent E. Giuliano

The paper provides clear summary descriptions of some fifteen ongoing language-processing research projects, and the descriptions are in sufficient depth of detail to give the general reader a good idea of what such systems are and how they are designed. Such factual state-of-the-art summaries are seen far too rarely in areas having to do with computer applications.

My main criticism of the paper relates to its failure to be critical of what is described. The paper might lead a casual reader to believe that considerable progress is being made—I tend instead to see evidence mainly of motion, with little real evidence of progress.

In presenting only the "tolerant viewpoint" of question answering, the paper constantly uses the expressions "syntactic analysis," "semantic analysis" and "logical analysis" suggesting that these phrases denote established methods of language processing requiring only engineering application. The brutal facts are as follows.

1. There is no experimental evidence to the effect that a substantial data base can be organized in a form suitable for subsequent "semantic analysis" except in the case of very restricted and almost trivial subject areas—family kinship relationships, baseball scores, simple geometrical figures, etc.

2. Beyond those trivial subject areas, only the foggiest understanding of the general problem of semantics exists, and research in this area is apt to require many years for real progress to be made.

3. Automatic syntactic analysis in its present state of development, although feasible, leaves the important problem of ambiguity to be resolved. Using a reasonably complete grammar, a typical sentence may lead to anywhere from a half-dozen to a hundred distinct analyses, and at present there is no way of selecting the correct one.

4. No clearly understood relation exists between meaning and logical formalism; that is, procedures for extracting meaning from test and expressing it in machine-manipulable notation are far from being fully understood. Moreover, while logical calculi are adequate for dealing with truth and falsehood, they account only inadequately for relevance, which is an independent notion since a statement may be false but yet relevant to a question.

In attempting to unify his descriptions of the various systems, Simmons often describes rather arbitrarily heuristic procedures as "principles." The notion seems to be that a procedure which is used in several computer programmed systems becomes a principle merely through use—despite the fact that its utility in any of the systems (or, as a matter of fact, the utility of such a system itself) remains unevaluated.

The idea that question answering should proceed through the three stages of syntactic analysis, semantic analysis and logical analysis is reminiscent of an idea, rather universally accepted in

1957, that machine translation of languages should proceed through the stages of lexical, syntactic and semantic processing. To date, however, such procedures have not been carried through successfully. The paper therefore tends to create an image of a developing unified science of question-answering systems—when as yet no such science exists in fact. The systems described are computer demonstrations; they have not, insofar as has been reported, been used for scientific experiments on language processing. That is, as reported in Simmons' paper, few attempts, if any, have been made to formulate and then test hypotheses about natural language for question-answering. Conclusions cannot be stated simply because hypotheses were not stated in a form suitable for testing in the first place. The result is that, while a number of such systems are being programmed, it will probably not be possible to conclude anything significant from their operation. Such systems typically work well in some instances, not so well in others, and the results are either uninterpretable or such that they could have been predicted without programming in the first place.

In summary, my reaction is that in a rush to demonstrate that question-answering can be done by computer, sight has too often been lost of the fact that much is yet to be learned about language, and that a demonstration can be only as good as the knowledge of language that goes into it. The existence of procedures of alchemy do not create a science—theories are needed which lead to testable hypotheses, and artifacts of computer usage are likely to be of utility only insofar as they are based on such theories or hypotheses.

●

REPLY TO COMMENTS

By R. F. Simmons

In the main Giuliano's remarks are an additional warning to the unwary reader that question-answering systems are in their infancy and that computational linguistics in this area (as in all others) falls far short of being an applied science. With this I heartily agree.

I do argue, however, that important principles are emerging from these studies and that the researches cited are not mere "computer demonstrations" but truly scientific approaches to the study of language. Each computer experiment embodies a set of hypotheses about the structure of language. These hypotheses are explicit only in the statements of the program—but in this form they are far more testable than a verbal statement could be.

To the extent that a question-answering program succeeds on its own sample text, hypotheses tend to be validated. Principles of language processing emerge as a result of repeated successes of the same techniques used in different settings by different researchers. As in all approaches to science, we are vulnerable to the possibility of incorrectly interpreting our results or of prematurely generalizing them. But the results are no less interpretable than in any other experimental approach.

I have attempted to extract some principles. The reader may justifiably disagree with these. However, to say that nothing can be concluded from question-answering machines is to deny that a science builds upon its models—not all of which can or should be experiments in the nineteenth century mode.

As for theory, we eagerly apply what little is available from linguistics and logic, but theory often lags far behind model building and sometimes derives therefrom.

We are not alchemists in search of an elixir of life or a philosopher's stone; we are accumulating knowledge by the toughest kind of experimentation—that of building small, very complex models and testing their limits.

Our models have not yet shown us how to build a practical question-answering system. But they have shown what can be done with today's knowledge and have led us to attempts to expand and improve presently inadequate approaches to syntactic, semantic and logical analyses of language.

Contributions to the Communications of the ACM

The *Communications of the* ACM serves as a newsletter to members about the activities of the Association for Computing Machinery, and as a publication medium for original papers and other material of interest. Material intended for publication may be sent to the Editor-in-Chief, or directly to the Editor of the appropriate department. It will also be considered for the *Journal of the Association for Computing Machinery.*

Contents—Submissions should be relevant to the interests of the Association and may take the form of short contributions or original papers. Short contributions may be published as letters to the editor, or in the news and notices department. Papers should be reports on the results of research, or expositional or survey articles. Research papers are judged primarily on originality; expositional and survey articles are judged on their topicality, clarity and comprehensiveness. Contributions should conform to generally accepted practices for scientific papers with respect to style and organization.

Format—Manuscripts should be submitted in duplicate (the original on bond paper) and the text should be double spaced on *one side* of the paper. Typed manuscripts are preferred, but good reproductions of internal reports are acceptable. Authors' names should be given without titles or degrees. The name and address of the organization for which the work was carried out should be given. If the paper has previously been presented at a technical meeting, the date and sponsoring society should appear in a footnote off the title.

Synopses—Manuscripts should be accompanied by an author's synopsis of not more than 175 words, setting out the essential feature of the work. This synopsis should be intelligible in itself without reference to the paper and should not include reference numbers. The opening sentence should avoid repetition of the title and indicate the subjects covered.

Figures—Diagrams should be on white bond or drafting linen. Lettering should be done professionally with a Leroy ruler (or, if necessary, in *sharp black* typing, with carbon reproducible ribbon). Photographs should be glossy prints. The author's name and the figure number should appear on the back of each figure. On publication, figures will be reduced to 3½ in. wide.

Citations—(1) *References to items in periodicals:* These should take the form: author, title, journal, volume number, date, pages. For authors, last names are given first, even for multiple authors; if an editor, the author's name is followed by: (Ed.). The author's name always ends with a period, either the period which is the abbreviation for his initial, or a period for the purpose. The title has only the first word and proper names (or their derivatives) starting with capital letters, and it ends with a period. The date is given in parentheses. The preferred method for abbreviations is that recommended by the International Standards Organization. Example:

Jones, R. W., and Anthony, T. Programming routines for Boolean functions. *J. ACM 5* (May 1960), 5-19.

(2) *References to reports or proceedings:* Author(s) name(s) and title—same style as for references in periodicals. Source, report number, city, date.

(3) *References to books:* Author(s) (same style as to periodicals). Title—all principal words start with a capital letter, and the title is underlined so that it will be set in italics. Page or chapter references follow the title, then a period. Publisher, city, year.

(4) *In lengthy bibliographies*, entries must be arranged alphabetically according to authors or editors names, except for those items to which no names can be attached.

Copyright—If material submitted for publication has previously been copyrighted, appropriate releases should accompany the submission. Copyright notices will be inserted when reprinting such material. If the author wishes to reserve the copyright of a computer program, upon his request a copyright notice in his name will be included when the porgram is published.

Association for Computing Machinery 211 East 43rd St., New York 17, N. Y.