AN APPROACH TOWARD ANSWERING ENGLISH QUESTIONS FROM TEXT *

R. F. Simmons, J. F. Burger and R. E. Long

System Development Corporation
Santa Monica, California

INTRODUCTION

Research on question answering by Raphael, Black, and Elliott, and our own work on Pro­
synthex II has shown that question-answering algo­
rithms can be most easily written if the text source is in the form of simple, explicitly structured sets of subject-verb-nominal strings. Question-answering algorithms that have thus far been developed include word- and structure-matching operations and some few logical inference functions. All of the sys­
tems cited have in some fashion limited their input language to simple subject-verb-nominal strings, thus elimin­
ing many problems of syntactic analysis and providing a normalized form for language data.

Our approach to question answering from natural­language text requires that both text and question be normalized into standard subject-verb-nominal kernels. Determining whether a set of text kernels answers a question or not is a complex matter of applying meaning-preserving transformations, to dis­
cover if pairs of apparently unlike kernels are in fact synonymous. The essential feature of the re­
search reported here is an approach to question an­
swering that depends on the discovery of sets of equivalence operators that can determine whether or not two English language strings are meaning-pres­
serving paraphrases of each other.

AUTOMATIC DERIVATION OF KERNELES

Recently Kuno and Foster have developed algo­
rithms that can derive kernel sentences from English text that has been analyzed by the Harvard Syntactic Analyzer. Joshi and a group at the University of Pennsylvania have also developed what may prove to be programmable algorithms for kernelizing. None of these algorithms is in completely programmed form as yet, nor are the definitions of kernels that have been so far developed completely suited to question-answering.

We have constructed a system that produces a type of text-normalizing kernel that we believe is more satisfactory for the purpose of answering ques­tions from English text. We base our derivation of kernels on Chomsky's theory of deep syntactic structures of sentences. For each phrase marker headed #S# of the underlying deep structure of a sentence, the kernel maker is required to produce a kernel string. The kernel strings are of the form X₁ R X₂ where X₁ and X₂ are usually in the form of nouns but may be reference numbers to other

* The research reported in this paper was sponsored by the Advanced Research Projects Agency Information Processing Techniques Office and was monitored by the Electronic Systems Division, Air Force Systems Command, under contract AF 19(628)-5166 with the System Development Corporation.
kernels, adjectives, adverbs, or null, and where R is a relational term ranging from simple verbs such as be, have, run, to a complex verb-particle or verb-preposition combination such as be-of, run-off, fly-from, etc. A null term (symbolized N') in an X position is typified by the “someone” or “it” of an incomplete passive construction. In the case of certain sentences such as “John wants to swim the Channel” the X₂ position of the kernel #1, “John wants 2,” must refer to a second kernel #2, “John swims the channel.” It is convenient to treat X₁ terms in a similar fashion in such sentences as “That she will come is delightful.”

In addition to such referencing within kernels from one to another, each kernel is associated with a cross-referencing term that records and labels its relation to coordinate or immediately superordinate kernels of the sentence. The intent of the cross-referencing is to maintain a labeled tree structure for the set of kernels that has been extracted from the sentence. Labels include conjunctions, relative pronouns, and indications of adjectival, prepositional and other types of subordination. For example in the sentence “John or Mary went to the store that sold skis,” the following kernels with reference terms are derived:

1. John went-to the store ( or 2)
2. Mary went-to the store ( or 1)
3. Store sold skis (that 1, 2)

The referencing scheme is designed to allow for the complete reconstitution of the original sentence and its surface structure. The reference terms will be seen below to be required for some question-answering operations.

Underlying the kernel-maker we use PLP II, which is a parsing system that produces (among other things) an immediate constituent analysis for a wide range of complex English text. Where phrase structure rules are inadequate to produce the IC analysis (particularly as in questions) a limited number of transformational rules have been included in the grammar. The form and operation of the PLP II system is described in Burger, Long and Simmons. It is important in regard to the kernel maker to note that it produces multiple analyses for most sentences that are given to it. However these are each in the form of a single structural description as illustrated in Fig. 1. The fact that only single unique descriptions are passed on to the kernel maker greatly simplifies its task. Our concern in this research is not with resolving syntactic ambiguity but with deriving kernel strings from the surface structure description of sentences.

Our procedure for obtaining kernels from the surface SD (structural description) of a sentence is outlined below. Although there may be room for argument that kernels representing deep structures can be obtained directly from the surface SD of the sentence, we interpret the procedures used by Kuno, Foster, and Lieberman and our own experience with recent experiments in kernelization to indicate a strong affirmative to the notion.

1. Given a complete surface SD of a sentence as in Fig. 1(c), scan it to discover structural indices of kernels. A structural index (SI) for a kernel is a three-node structure A(B C) such that B and C are both nodes dominated by A. Thus NP(N PP), NP(NP PP), NP(Adj. N), S(NP VP) are all examples of structural indices that identify kernels. NP(Art. N), VP(V NP), VP(V PP) and PP(Prep. N) are examples that do not.

JACK AND JILL WENT UP THE HILL AND FETCHED A PAIL OF WATER.

(2 PARSINGS)

1
(1 (JACK (AND (JILL)))
(VP (V WENT) (PP (PREP UP) (NP (ART THE) (N HILL))))
(NP (ART A) (NP (ART A) (NP (ART (PP (PREP OF) (N WATER)))))))

(1) (A ART PAIL 11)

PAIL (IS OF) WATER

(12 OF (PREP PAIL 11) (13 WATER N OF 12))

(11 FETCHED)

WENT (UP (HILL (THE)))

(10 A ART PAIL 11)

Figure 1. Examples of computer-produced syntactic analysis from PLP II, showing simple kernels.
2. Look up the SI in a list of kernel structural indices (KSI) to discover if it is a KSI. If not, continue the scan. If it is a KSI, there will be associated with it a function that operates on the SD of the sentence to produce, first a kernel SD, then a kernel string. The functions are exactly equivalent to inverse transformational rules.

For example given the KSI, NP(Adj. N), the associated function transforms it to $S_K$ (N be Adj.) which is the SD of a kernel string. ($S_K$ stands for kernel sentence.) A more complicated case is illustrated by the KSI, NP(rel. pron. VP). Here the function is required to find the noun antecedent of the relative pronoun and substitute it into the resulting kernel string. This occurs in two steps, which may be most easily shown in a transformational notation:

$$\#1 \ NP((\ldots N_1 \ldots) (NP(\text{rel. pron.} + VP))) \rightarrow NP((\ldots N_1) (NP(N_1 + VP)))$$

$$\#2 \ NP(N_1 + VP) \rightarrow K_s (N + VP)$$

The $K_s$ is now able to generate a kernel string.

3. Label each kernel. The labels for each kernel correspond to specific features of the KSI functions that generate them. The label for the last illustration would include the actual relative pronoun. The label for a prepositional or adjectival kernel would be "prep." and "adj.,” respectively.

4. Reference each kernel to the kernel or kernels that are coordinate and superordinate to it. This information is available as a result of the order of processing from deepest structures upward and from the labels previously assigned.

So far we have experimented with LISP functions that accomplish steps (1) and (2) and, though we can see that (3) and (4) are not difficult in principle, we have not yet fixed on a particular algorithm. It will be noticed that we do not follow transformational programming conventions of the type adopted by the MITRE system or by Foster. The reason is that in LISP notation, the SD of a sentence is relatively easy to deal with directly and with generality. Chomsky, the MITRE group, Foster and our own experience indicate that relatively few KSIs and functions will be required to generate kernels from a wide range of English sentences.

Serious problems are expected in recovering kernels from those surface structures that have resulted from the repeated application of deletion transformations. Additional problems are expected in developing fully adequate notation for cross-referencing and labeling the kernels from a sentence. Problems involving the need to find antecedents for pronouns and other pronominal expressions will probably also have to be dealt with. In this latter regard rules for resolving anaphora developed by Olney and Londe have proved helpful.

Other problems concern our definition of the R (relational) term of the kernel—we are by no means certain that our method of dealing with verbs, adverbs, and prepositions is the best. We expect further light to be shed on this problem as a result of attempting to use the kernels for answering questions.

ANALYSIS OF QUESTIONS

Our basic approach to question answering is to make a surface structure analysis of the question, resolve it into kernels, and then compare the kernels resulting from the question with those stored in a directed graph structure. The stored data are obtained by kernelization of English sentences that have been given to the system. As in the approach to kernelizing above, PLP II is used to obtain a surface parsing of the question.

The comparison of Question kernels (Q-kernels) with Text kernels (T-kernels) will be a simple comparison only in the case that the question is complete, that is, of the form "Aux V NP VP," or "is NP NP," (e.g., do NP VP, can NP VP, is NP NP), and there exists an exactly corresponding T-kernel. Generally, the problem will be one of discovering if an appropriate set of T-kernels can be found in the data structure and transformed into a match with the set of Q-kernels derived from the question.

In perhaps the simplest case, the question “Is X a Y?” is first resolved into the Q-kernel, X be Y. X and Y are used to search the headings of the data

* This approach was discussed earlier by Walker and Bartlett.4
† This discussion assumes “N be N” kernels to select the class membership sense of be. Kernels of the type “N be Adj.,” follow a different line of logic.

From the collection of the Computer History Museum (www.computerhistory.org)
structure (described in Simmons'). If the T-kernel X be Y is discovered there is no problem. If the T-kernels X be W, W be V, B be Y are discovered, the fact that "be" is known to be logically transitive allows X be Y to be derived, again answering the question. This logic applies also for such obviously transitive relations as farther than, nearer than, brother of, part of, etc. (see Raphael,1 Cooper,14 and Black 3).

If no transitive relation is found to lead from X be W to X be Y, the synonym operation of inferring that W ≡ Y may show that (X be W) → (X be Y), and again an answer is possible. The operations illustrated in this simple question are almost the only ones that have been used in question-answering systems up to the present.8 They are obviously not sufficient to answer more than a tiny subset of questions. The more ambitious systems 16,17 have devoted a great deal of effort to dealing with analysis of the question into its request structure, its syntactic structure, finding word and structure matches, using synonyms, etc., but they have dealt only superficially with what we see as the central problem of transforming from a set of relevant T-kernels † into the exact form of the Q-kernels.

More generally the problem can be described as follows:

Given the set of
Question kernels, Q1 = {KQ1, KQ2 ... KQn}

and the set of Text kernels, T1 = {KT1, KT2 ... KTm}

If there is a set of transforming operators, TO = {TO1, TO2 ... TOk}

such that, Q1 = TO × T1, or, T1 = TO × Q1,

then Q1 has a complete answer. ‡

Discovering operators of the set TO and the conditions under which they may be applied to English words appears to be the basic research problem in question answering.

In Belnap's 12 logic of questions, a question may be analyzed into two parts, a request and a set of alternatives. The request can usually be recognized in such forms as "what," "what (noun)," "what are (number)," "where," "when," "how," "how many," etc. The set of alternatives may be the remainder after the request kernels are removed.

Given a question: "What are the paths of rockets or missiles called?" the following set of Q-kernels can be derived:

#1) Paths are what?
#2) Someone calls paths.
#3) Paths are-of rockets.
#4) Paths are-of missiles.

There is obviously a request kernel, #1), and two kernels specifying the set of alternatives. (Whether or not it will always be as easy to separate request kernels and alternatives is not known but it is doubtful.) The alternatives are used in a question-answering system as index terms to find relevant information. The request is used to evaluate the information (e.g., "how many" requires a number, "where" requires a place coding, "when" a time coding, etc.) and sometimes also to process it (how many, or the commands, "list 7, name 3, etc." require counting or listing functions). 8

Although both request kernels and alternative kernels must be identified so that we can understand how to use them, the main difference seems to be that the request kernel definitely indicates a set of operations to be performed on the answer kernels while the alternative kernels imply a set of operations to be performed on the data base. It may be that these operations may be conceived of in the same framework as the transforming operations required to match sets of T-kernels to a set of Q-kernels (with a consequent simplification of the whole system).

LINGUISTIC INFERENCE VIA EQUIVALENCE-OPERATORS

We see the basic problem of question answering to be one of discovering equivalence-operators and describing the conditions under which they can be applied to make Q- and T-kernels identical. Chomsky (p. 162), 8 among others, offers the following two sets of sentence-pairs as examples of paraphrases that are not accounted for by identical deep syntactic structures:

* In this research we are concerned only with fact retrieval questions and are ignoring the algebra problem or questions typified by "How many letters are in Oliver Goldsmith?"
ANSWERING ENGLISH QUESTIONS FROM TEXT

1. John strikes me as pompous.
   I regard John as pompous.
2. I liked the play.
   The play pleased me.

The second pair is in the form of kernels and it can be seen that some form of inverse operator * can be applied to account for the paraphrase:

\[(N_1 \text{ likes } N_2) \times TO_i = (N_2 \text{ pleases } N_1)\]

where: “\(\times\)” signifies the application of the operator, and “\(\equiv\)” is interpreted to mean “functionally equivalent in context.” The operation appears to apply for kernels involving “like” as a verb in the syntactic context of nouns.

In set 1, however, the situation appears to be more complex. In the first place it applies to a complex three-part relationship as follows:†

\[(N_1 \text{ strikes } N_2 \text{ as Adj.}) \times TO_{i1} = (N_2 \text{ regards } N_1 \text{ as Adj.})\]

This equivalence operation appears to apply without exception with reference to strikes-as: regards-as but it does not apply to the kernel \((N_1 \text{ strikes } N_2)\) especially where the striking is accomplished with a club.

Apparently the inverse-equivalence operator is a very common occurrence in English in such pairs as bought-sold, gave-received, read-write, etc., as well as in the active-passive syntactic transformation. There appear to be many cases of its application where only syntactic conditions restrict its use.

Another frequent case of paraphrase that has proved important in question answering is the matter of substituting synonyms. If we consider a synonym-equivalence operator, \(T_s\), there are some cases such as:

\[(N_1 \text{ eats } N_2) \times T_s = (N_1 \text{ devours } N_2)\]

where the equivalence may be sufficient for practical purposes but where there are probably always semantic restrictions limiting the substitution. Sparck Jones 18 discusses this point in detail.

There is also a large set of weak implication operators that may be applied to English statements but apparently only under highly specified semantic and pragmatic conditions. For examples:

\[*\text{Inverse } (R_1R_2) = V_{x,y} (x R_1 y \rightarrow y R_2 x)\]

† It is of interest to note how easily this applies to a five-part nonkernel in contrast to the cumbersomeness of applying it to two kernels.

The question marks and the symbol “\(\pm\)” indicate our ignorance of the conditions under which these equivalences may be valid.

A more satisfying set of operations has to do with class membership. If \(A\) is a member of the class \(B\), \(A \rightarrow B\).

Thus:

- walk \(\rightarrow\) move
- go \(\rightarrow\) move
- aardvark \(\rightarrow\) mammal
- mammal \(\rightarrow\) animal
- steak \(\rightarrow\) meat
- meat \(\rightarrow\) food

However such relationships are not easy to discover for many if not most English words.

From our work in question answering we are dimly aware that there are many complicated equivalence operations involved in the various ways that answers to a given question may be paraphrased. The matter of matching Q-kernels to T-kernels will prove a very complex process requiring not only the discovery of appropriate equivalence operators and the syntactic and semantic conditions of their use, but also an understanding of the logical properties (e.g., transitivity, symmetry, asymmetry, reflexiveness, etc.) of relation terms to further define their application. In our work we concentrate mainly on the discovery of operations that depend only on syntactic restrictions and believe that these may prove to be sufficient to greatly increase our understanding of the question-answering process.

PROPOSED DISCOVERY PROCEDURE FOR EQUIVALENCE-OPERATORS

Our approach to the study is to select a broad range of question types with varying forms of answering statements associated to each question. Questions and answers will be reduced to kernels (by hand if necessary or by the kernelizer described above). We will attempt to discover a set of equivalence-operators that can transform the T-kernels into Q-kernels. Such operators will be tested on larger samples of English text by being translated into LISP functions and tested in the context of Protosynthex II.
It is our hope that the set of equivalence operators that are discovered and the LISP functions that embed them may eventually form the basis of a powerful question-answering and paraphrasing language.

The questions will be selected from the Modern Science Quiz Book, which offers a wide range of syntactic forms and semantic content. Although we do not propose any rigidly specified sampling procedure, we intend to pay particular attention to the more difficult and complicated questions represented in this compendium.

In selecting answer sets for each question, we will depend on encyclopedias such as Compton's and the Golden Book to insure a wide and varied range of paraphrases. We expect also to compose answers in some cases, and to use answers made up of more than one sentence.

Table 1 shows a question Q1, the set of its Q-kernels and two alternate answers A1 and A2 and the sets of T-kernels derived from them. We will describe some operations that can be applied to the Q-kernels and to the T-kernels that can serve to make them isomorphic. Since this theory of question answering has not yet been empirically tested, we do not pretend that this discussion or the resulting operators are more than illustrative—they are included only to exemplify the line of approach.

The kernel Q1.1 may first be subject to a deletion operator. The basis for this is that "N' calls N" is a kernel (where N' is deleted after the passive transformation) which shows that "calls" is used in a meta-structure that is not necessarily relevant to answering the question. After this operation the kernels A2.1 and A2.3 form a sufficient answer (assuming "is = are" and recognition of the meaning of the "or" relation between Q1.2 and Q1.3). The resulting answer is in the form "Paths are trajectories, paths are of missiles," or some combination of these into a more complex sentence.

To discover how A1 is also an answer is more difficult. As native speakers of English we understand that in kernel A1.1 it is legitimate to transform "rockets leave paths" into "paths are of rockets"; on the other hand if the kernel were "rockets leave planets" it is not at all satisfactory to say "planets are of rockets." Thus, if it is possible to transform "N1 leave N2" into "N2 are of N1," there must be semantic conditions limiting the N1-N2 pairs to which equivalence applies.

Ignoring this possibility, kernel A1.3 appears more tractable. "N1 are-as N2" appears to lend itself invariably by deletion operation, Tas, into "N1 are N2." (Reasons for this have been found in considering "as" in certain environments as a marker of discourse equivalence by Olney (unpublished).

By using:

\[ T_{as} \times (N1 \text{ are-as } N2) \rightarrow (N1 \text{ are } N2), \]

kernel A1.3 now states "Paths are trajectories." Since "is" has the property of transitivity, it follows that we can equate "trajectories" and "paths" in A1.4 to get a new kernel A1.5 "paths are of rockets." Now a match is available as follows:

\[
\begin{align*}
Q1.1 & \quad (N' \text{ calls paths}) = \phi \\
Q1.2 & \quad (\text{Paths are what}) = A1.3 T_{as} = \\
Q1.3 & \quad (\text{Paths are-of rockets}) = A1.5 = \\
& \quad (\text{paths are trajectories})
\end{align*}
\]

The discussion essentially illustrates our approach to a discovery procedure for finding equivalence operators. The importance of translating these into programs and testing them on larger samples of text can hardly be overrated. In the first place, the need to program them insures a complete understanding of what is the operational meaning of such easy phrases as "equating 'trajectory' and 'path' on the

---

*This assertion must be worked out in more detail for "calls," "names," etc.
ANSWERING ENGLISH QUESTIONS FROM TEXT

In the second place the program makes them more easily testable against larger sets of text so that a fair assurance can be developed that a final set of equivalence operators is of wide generality.

In summary, we propose to test the theory by developing sets of equivalence rules that can be used to transform T-kernels from statements that are answers to questions into the form of the Q-kernels from the questions. As these rules are developed, they will be expressed in the form of LISP functions and used in the context of the Protosynthex II question-answering system to test their validity on a wider range of questions.

ACKNOWLEDGMENTS

We are grateful to Dave Londe and John Olney for their critical reading and helpful suggestions and to Bruce Fraser, whose critical comments on a first draft of this paper were extremely helpful.

REFERENCES


* e.g., Transitivity of R = Vx,y,z (x R y & y R z → x R z).


