Operating Statistics for The Transformational Question Answering System

Fred J. Damerau

Mathematical Sciences Department
IBM Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

This paper presents a statistical summary of the use of the Transformational Question Answering (TQA) system by the City of White Plains Planning Department during the year 1978. A complete record of the 788 questions submitted to the system that year is included, as are separate listings of some of the problem inputs. Tables summarizing the performance of the system are also included and discussed. In general, performance of the system was sufficiently good that we believe that the approach being followed is a viable one, and are continuing to develop and extend the system.

Introduction

Natural language question answering systems have been the subject of much research over approximately 20 years. In only very few cases have such systems been exposed to real users trying to solve real problems, another example perhaps being Krause (1979). In an attempt to see if a useful natural language query system could be built for an application which existed independently of the research program, an approach was made to the Planning Department of the City of White Plains asking them to take part in such an experiment. Their incentive was the free use of an interactive query facility which would allow them to explore their data base more freely than the batch computer facility run by the city could do. The remainder of the paper describes the user environment, describes the operation of the TQA system, and discusses the operating results in the first year of operation, 1978.

The User Environment

When the experiment was first being discussed, the Planning Department had five professionals plus the services of a consultant more or less constantly available. The main files used for planning were the *parcel file* and the *geobase file,* which had been converted to machine readable form under a grant from the Department of Housing and Urban Development. Department members were very familiar with these files, in many instances knowing the correspondence between codes in the file and their English equivalents.

The parcel file contained a record for each parcel of land in the city, approximately 10,000 of them, which had a taxable existence. Each record contained the account number, the block, the owner, the address, land use information, number of dwelling units, area, taxes, and the like. The geobase file contained a record for each city block, telling what census tract it was in, what traffic zone, what neighborhood association, etc. Selected summaries of these files had been prepared by the City's computing center, and the files themselves had been printed in a couple of large volumes. Ad hoc queries required special programs to be written by the computing center, with sufficient delay that this was seldom done. Thus, we were told that during the 1974 gasoline shortage, the parcel file was searched by hand on the land use code field to find the locations of all the gas stations so that police could be routed there to direct traffic. Other uses of the files are apparent from inspection of the questions asked of the system, a sample of which are given in Figure 1.

Although the terminal was located in an open area available to all members of the department, it turned out that most of the questions were asked by one of the department members, who had been designated as our liaison. The reason for this was never clear, but may have simply been normal reluctance to experiment with radically new technology on the part of the older members of the department. Other personnel did sometimes use the system, but under his sign-on code,

0362-613X/81/010030-13$01.00

```
Where are the parcels with a LUC of 641 ?
What single family houses in Fisher Hill have
    exemptions greater than $0 ?
How many two family houses are there in the
    Oak Ridge Residents Assn. ?
Who owns the parcels in subplanning area 7.60 ?
Where are the churches on North Street ?
What parcels on Stevens St. have a LUC of 116 ?
What parcel in ward 1 does city own ?
Print the parcel area of the LUC'S 300 - 399 !
What is the assessment of the parcels in the
    Battle Hill Assn. having more than 3 dwelling units ?
Where is Martin 's parcel ?
Where are the apartment dwellings which have more than 50
    units which are more than 6 stories high on Lake St. ?
Where is the Calvary Baptist Church parcels ?
Where is parcel 50550006401 ?
What properties does Longo own ?
Where are the apartment buildings having less
    than 8401 sq ft ground floor area  ?
```

Figure 1. Example inputs to TQA.

so that the actual user for any question cannot be determined from the log in most cases. During the course of the experiment, a new planning director was appointed, who changed the mission of the department somewhat. The result was that members of the planning department did fewer of the conventional planning activities than formerly, with results that are apparent in the statistics that follow.

The TQA System

The TQA system, originally named REQUEST, has been under development for some time at the IBM Thomas J. Watson Research Center. An experimental application using business statistics had been quite satisfying. For the White Plains application, major additions were made to the lexicon, new grammar rules extending coverage were written, a new component for interfacing the grammar and a data base was developed, and an interface was built to an existing data base management system, the RSS. The system was installed in White Plains late in 1977 for final debugging, and turned over to the planners at the beginning of 1978. It was disconnected at the end of 1979 partly for legal reasons and partly because we felt little new could be learned by leaving it there.

A generalized flow diagram of the TQA system is given in Figure 2, and an example of processing in Figures 3a-g. The structures printed in Figure 3a are a bracketed terminal string representation of structures which are stored and manipulated as trees by the processing programs. The trees, together with their associated complex features, for the example are shown in Figures 3b-g. These structures are the outputs at similarly named points of the flow diagram in Figure 2.

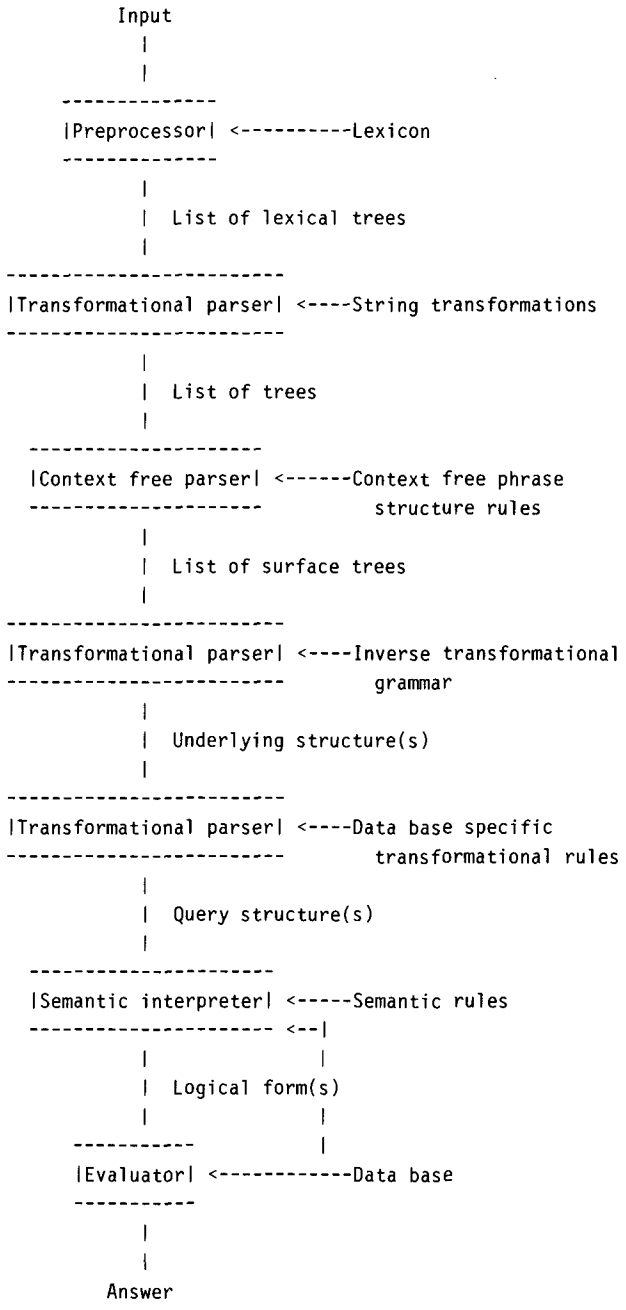

Figure 2. Flow diagram of TQA.

Input from an IBM 3275 display station is fed to the preprocessor, which segments the input character string into words and performs lexical lookup. The process of lookup is complicated somewhat by a provision for synonym and phrase replacement. Words like "car" and "automobile" are changed to "auto", and strings like "gas station" are frozen into single lexical units. The output from the lexical lookup is a list of trees, each tree containing part-of-speech information, syntactic features and semantic features, as required. A description of the lexical component, now obsolete

```
TYPE NEXT QUESTION.
Where are the gas stations in traffic zone 579 ?
 STRING TRANSFORMATIONS:
((AT ((WH SOME) (PLACE X11))) BE THE
((GAS_STATION =S17) X4) IN (TRAFFIC_ZONE 579) ?)
 SURFACE STRUCTURES:
1. ((AT ((WH SOME) (PLACE X11))) BE (THE
(((GAS_STATION =S17) X4) (IN (TRAFFIC_ZONE 579)))) ?)
2. ((AT ((WH SOME) (PLACE X11))) BE (THE
((GAS_STATION =S17) X4)) (IN (TRAFFIC_ZONE 579)) ?)
 UNDERLYING STRUCTURES:
1. (BD LOCATED (THE (((GAS_STATION =S17) X4)
(* BD LOCATED X4 (TRAFFIC_ZONE 579) BD *)))
((WH SOME) (PLACE X11)) BD)
 QUERY STRUCTURES:
1. (BD ADDRESS ((WH SOME) (THING X11))
(THE (((GAS_STATION 553) X4)
(* BD TRAFFIC_ZONE 579 X4 BD *))) BD)
 LOGICAL FORM:
(setx 'X11
  '(foratleast @1 'X63
    (setx 'X4
      '(and
        (testfct '@579
          '('TRAFZ X4 '@1976)
          '= )
        (testfct 'O553
          '('LUC X4 '@1976)
          '= ) ) )
    '(testfct X11
      '('ADDRESS X63 '@1976)
      '= ) ) )
 ANSWER:
                              ADDRESS
 1976
  2 06300 02000          122 S LEXINGTON AV
  2 06600 00100          101 W POST RD
  2 05500 09300          102 W POST RD
  2 07100 03300          109 W POST RD
  2 07100 02900          115 W POST RD
```

Figure 3a. Short trace of example question showing major intermediate structures.

in its detail but still valid in main outline, is given in Robinson (1973). Without going into great detail, one can see in Figure 3b that "gas station" and "traffic zone" have been made into single units. The node *=S17* in the entry for gas station is a macro standing for a bundle of semantic features. Many of the feature names should be obvious, but, e.g., *PL* stands for "place", *PAG* for parcel aggregate, i.e., an aggregate of separate parcels, and *CINS* for "cardinal insertion', i.e., can be followed by a cardinal number.

```
( (where
    ( ((RP ((+ WH) (+ LOC)))
        ((NOM)
          ((NOUN ((+ REL) (- HU) (+ PL)))
            ((INDEX ((- CONST)))
              ((XO)) ) ) ) ) ) )
  (are
    ( ((BAUX ((- PAST) (- SG)))
        ((BE)) ) ) )
  (the
    (((DET) ((THE)))) )
  (gas_station
    ( ((NOM)
        ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
          ((V)
            ((V) ((GAS_STATION)))
            ((=S17)) )
          ((INDEX ((- CONST)))) ) ) ) )
  (in
    (((PREP) ((IN)))) )
  (traffic_zone
    ( ((NOM)
        ((NOUN ((- HU) (+ SG) (+ PL) (+ PAG) (+ GEO)))
          ((V ((+ OGEN) (+ POBJ)))
            ((TRAFFIC_ZONE)) )
          ((INDEX ((- CONST) (+ CINS)))) ) ) ) )
  (579
    ( ((VADJ ((+ ADJ) (+ CARD) (+ D3)))
        ((579)) ) ) )
  (?
    ( ((PUNCT ((+ QUES)))
        ((?)) ) ) ) )
```

Figure 3b. List of lexical trees.

The list of lexical trees is input to a set of *string transformations*, described in Plath (1974). These transformations operate on adjacent lexical items to deal with patterns of classifiers, ordinal numbers, stranded prepositions, and the like. The effect of this phase is to reduce the number of surface parses and the amount of work done in the transformational cycle. Referring to Figure 3c, note that "579" has been incorporated with "traffic zone" under a single node, *PROPNOM*.

The resulting list of trees is input to a context free parser, which produces a set of *surface trees*. In the example, two surface trees are produced, shown in Figures 3d and 3e. The trees differ in the point of attachment of the phrase "in traffic zone 579". In structure 1, it is attached to the NP "the gas station", and in structure 2 it is directly under the *S* node.

The recognizer attempts to find an *underlying structure* for each surface tree (Plath 1973, 1976). Typically, only one of a set of surface trees will result in an underlying structure. In the example, the struc-

```
((AT ((WH SOME) (PLACE X11))) BE THE
((GAS_STATION =S17) X4) IN (TRAFFIC_ZONE 579) ?)
((ST)
  ((PP)
    ((NSPREP) ((AT)))
    ((NP)
      ((NOM)
        ((V ((+ ADJ) (+ QUANT)))
          ((WH))
          ((SOME)) )
        ((NOM)
          ((NOUN ((+ SG) (- HU) (+ PL)))
            ((V) ((PLACE)))
            ((INDEX ((- CONST)))
              ((X11)) ) ) ) ) ) )
  ((BAUX ((- PAST) (- SG)))
    ((BE)) )
  ((DET) ((THE)))
  ((NOM)
    ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
      ((V)
        ((V) ((GAS_STATION)))
        ((=S17)) )
      ((INDEX ((- CONST)))
        ((X4)) ) ) )
  ((NSPREP) ((IN)))
  ((PROPNOM)
    ((NOUN ((- HU) (+ SG) (+ PL) (+ PAG) (+ GEO)))
      ((V ((+ OGEN) (+ POBJ)))
        ((TRAFFIC_ZONE)) )
      ((INDEX ((+ CONST)))
        ((VADJ ((+ ADJ) (+ CARD) (+ D3)))
          ((579)) ) ) ) )
  ((PUNCT ((+ QUES)))
    ((?)) ) )
```

Figure 3c. List of trees after string transformations.

ture in which the prepositional phrase was attached to "gas station" survives. The underlying structures are similar to those proposed under the variant of transformational grammar called *generative semantics*. This is not the place to defend that particular theory or its use on the TQA system; suffice it to say that a considerable body of grammatical work on English has been done in a style compatible with this theory. To simplify, every *S* in the underlying structure has a predicate and some number of noun phrase arguments. The noun phrases may dominate imbedded S's. In the example, the top level predicate is LOCATED with arguments of "gas station" and "some place". The NP for "gas station" dominates an S which also has a main predicate of LOCATED and arguments of "X4", i.e., the same index as that for "gas station", and "traffic zone 579", i.e., the gas station located in traffic zone 579. Notice that the parser has supplied both

```
1. ((AT ((WH SOME) (PLACE X11))) BE (THE
(((GAS_STATION =S17) X4) (IN (TRAFFIC_ZONE 579)))) ?)
((S1)
  ((PP)
    ((NSPREP) ((AT)))
    ((NP)
      ((NOM)
        ((V ((+ ADJ) (+ QUANT)))
          ((WH))
          ((SOME)) )
        ((NOM)
          ((NOUN ((+ SG) (- HU) (+ PL)))
            ((V) ((PLACE)))
            ((INDEX ((- CONST)))
              ((X11)) ) ) ) ) ) )
  ((BAUX ((- PAST) (- SG)))
    ((BE)) )
  ((NP)
    ((DETX)
      ((DET) ((THE))) )
    ((NOMX)
      ((NOMZ)
        ((NOM)
          ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
            ((V)
              ((V) ((GAS_STATION)))
              ((=S17)) )
            ((INDEX ((- CONST)))
              ((X4)) ) ) )
        ((Z1)
          ((PP3X)
            ((PP)
              ((NSPREP) ((IN)))
              ((RPX)
                ((NP)
                  ((PROPNOM)
                    ((NOUN ((- HU) (+ SG) (+ PL)
                            (+ PAG) (+ GEO)))
                      ((V ((+ OGEN) (+ POBJ)))
                        ((TRAFFIC_ZONE)) )
                      ((INDEX ((+ CONST)))
                        ((VADJ ((+ ADJ) (+ CARD) (+ D3)))
                          ((579)) ) ) ) ) ) ) ) ) ) ) ) ) )
  ((PUNCT ((+ QUES)))
    ((?)) ) )
```

Figure 3d. Surface structure tree 1.

instances of "located". A fuller explanation is given in Plath (1973, 1976).

The data base has no predicate, i.e., column heading, named "located". Even if it did, the two LOCATEDs in the underlying structures are different, one for address and one for traffic zone. The underlying structure is designed to reflect the linguistic structure of the query, but, as in this case, that structure

```
2. ((AT ((WH SOME) (PLACE X11))) BE (THE
((GAS_STATION =S17) X4)) (IN (TRAFFIC_ZONE 579)) ?)
((S1)
  ((PP)
    ((NSPREP) ((AT)))
    ((NP)
      ((NOM)
        ((V ((+ ADJ) (+ QUANT)))
          ((WH))
          ((SOME)) )
        ((NOM)
          ((NOUN ((+ SG) (- HU) (+ PL)))
            ((V) ((PLACE)))
            ((INDEX ((- CONST)))
              ((X11)) ) ) ) ) ) )
  ((BAUX ((- PAST) (- SG)))
    ((BE)) )
  ((NP)
    ((DETX)
      ((DET) ((THE))) )
    ((NOMX)
      ((NOM)
        ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
          ((V)
            ((V) ((GAS_STATION)))
            ((=S17)) )
          ((INDEX ((- CONST)))
            ((X4)) ) ) ) ) )
  ((PP4X)
    ((PP)
      ((NSPREP) ((IN)))
      ((RPX)
        ((NP)
          ((PROPNOM)
            ((NOUN ((- HU) (+ SG) (+ PL) (+ PAG) (+ GEO)))
              ((V ((+ OGEN) (+ POBJ)))
                ((TRAFFIC_ZONE)) )
              ((INDEX ((+ CONST)))
                ((VADJ ((+ ADJ) (+ CARD) (+ D3)))
                  ((579)) ) ) ) ) ) ) ) )
  ((S2PCX)
    ((PUNCT ((+ QUES)))
      ((?)) ) ) )
```

Figure 3e. Surface structure tree 2.

may not match a particular data base organization. There are many approaches one could take to make this match. We have chosen to implement the matching function as a separate transformational component in the grammar (Damerau 1977). The underlying structure itself is therefore input to the transformational recognizer, using a (small) set of grammar rules tailored to a specific data base and produces a *query structure*. Query structures are similar to underlying structures in form, but reflect the particular meaning

```
1. (BD LOCATED (THE (((GAS_STATION =S17) X4)
(* BD LOCATED X4 (TRAFFIC_ZONE 579) BD *)))
((WH SOME) (PLACE X11)) BD)
((S1 ((- PAST) (+ WH) (+ QUES) (+ TOP)))
  ((BD))
  ((V ((+ ADJ) (+ EN) (+ LOC) (+ TEMP)
        (+ PSUBJ) (+ POBJ)))
    ((LOCATED)) )
  ((NP)
    ((DET) ((THE)))
    ((NOM)
      ((NOM)
        ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
          ((V)
            ((V) ((GAS_STATION)))
            ((=S17)) )
          ((INDEX ((- CONST)))
            ((X4)) ) ) )
      ((S1)
        ((BD))
        ((V ((+ ADJ) (+ EN) (+ LOC) (+ TEMP)
              (+ PSUBJ) (+ POBJ)))
          ((LOCATED)) )
        ((NP)
          ((NOM)
            ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
              ((INDEX ((- CONST)))
                ((X4)) ) ) )
          ((NP ((+ IN) (+ LOC)))
            ((NOM)
              ((NOUN ((- HU) (+ SG) (+ PL)
                      (+ PAG) (+ GEO)))
                ((V ((+ OGEN) (+ POBJ)))
                  ((TRAFFIC_ZONE)) )
                ((INDEX ((+ CONST)))
                  ((V ((+ ADJ) (+ CARD) (+ D3)))
                    ((579)) ) ) ) ) ) )
        ((BD)) ) ) )
  ((NP ((+ AT) (+ LOC)))
    ((NOM)
      ((V ((+ ADJ) (+ QUANT)))
        ((WH))
        ((SOME)) )
      ((NOM)
        ((NOUN ((+ SG) (- HU) (+ PL)))
          ((V) ((PLACE)))
          ((INDEX ((- CONST)))
            ((X11)) ) ) ) ) )
  ((BD)) )
```

Figure 3f. Underlying structure.

constraints resulting from the format and content of a given data base. In Figure 3g, one can see that the first instance of LOCATED has been changed to the

```
1. (BD ADDRESS ((WH SOME) (THING X11))
(THE (((GAS_STATION 553) X4)
(* BD TRAFFIC_ZONE 579 X4 BD *))) BD)
((S1 ((- PAST) (+ WH) (+ QUES) (+ TOP)))
  ((BD))
  ((V ((+ ADJ) (+ EN) (+ LOC) (+ TEMP)
          (+ PSUBJ) (+ POBJ)))
    ((ADDRESS)) )
  ((NP ((+ AT) (+ LOC)))
    ((NOM)
      ((V ((+ ADJ) (+ QUANT)))
        ((WH))
        ((SOME)) )
      ((NOM)
        ((NOUN ((+ SG) (- HU) (+ PL)))
          ((V) ((THING)))
          ((INDEX ((- CONST)))
            ((X11)) ) ) ) ) )
  ((NP)
    ((DET) ((THE)))
    ((NOM)
      ((NOM)
        ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
          ((V)
            ((V) ((GAS_STATION)))
            ((LUC) ((553))) )
          ((INDEX ((- CONST)))
            ((X4)) ) ) )
      ((S1)
        ((BD))
        ((V ((+ ADJ) (+ EN) (+ LOC) (+ TEMP)
              (+ PSUBJ) (+ POBJ)))
          ((TRAFFIC_ZONE)) )
        ((NP ((+ IN) (+ LOC)))
          ((NOM)
            ((NOUN ((- HU) (+ SG) (+ PL)
                      (+ PAG) (+ GEO)))
              ((INDEX ((+ CONST)))
                ((V ((+ ADJ) (+ CARD) (+ D3)))
                  ((579)) ) ) ) ) )
        ((NP)
          ((NOM)
            ((NOUN ((- HU) (- SG) (+ PL) (+ SV)))
              ((INDEX ((- CONST)))
                ((X4)) ) ) )
          ((BD)) ) ) )
  ((BD)) )
```

Figure 3g. Query structure.

predicate *ADDRESS* and the second instance to the predicate *TRAFFIC_ZONE*.

The query structure tree is processed by a Knuth-style semantic interpreter (Petrick 1977), producing a *logical form*. A logical form can best be thought of, in our context, as a retrieval expression which is to be evaluated, producing an answer to the English input query. Referring to Figure 3a, the logical form can be read as:

> Find the set of things X11 such that for at least 1 of the things X63 in the set of things X4 where the traffic zone of X4 is 579 and the LUC (land use code) is 0553, (viz., the set of account numbers having both these properties), the address of X63 is X11.

The process of answer extraction from the data base is accomplished by a combination of LISP and PL/I programs (Damerau 1978), and an experimental relational data base management system called Relational Storage System (RSS) (Astrahan, et al., 1976). The RSS provides the capability to generate a data base of n-ary relations, with indexes on any field of the relation, and low-level access commands like OPEN, NEXT, CLOSE, with appropriate parameters, to retrieve information from such a data base. This particular data base had just one relation of 40 columns. The LISP programs examine the logical form to establish relationships between variables and to generate requests to the data base component to find items with specified properties. In the example, one retrieval request would find the qualifying account numbers, i.e., the X4s, and a second request would find the addresses of those account numbers.

Notice that the logical form simply specifies a set of addresses as the answer. This is clearly unsatisfactory, and the data base interface program supplies the account number for each address as part of the answer. The long numbers in the answer are the parcel identifiers, (ward-block-lot) referred to above as account numbers and sometimes called lot numbers.

All the processing modules are under the control of a driver module which maintains communication with the user, calls the processors in the correct sequence, and tests for errors.

Usage Statistics

The statistics presented below are not based on a constant system. When a problem was discovered in the course of operation, an attempt was usually made to change the system so as to make the problem query run successfully. This was not always possible, but a great many changes were incorporated into the system during the course of the year. An attempt to compensate in part for the effect of this situation on the statistics was made by rerunning all the sentences which failed during the year with the system in use in May, 1979. The results of this run are incorporated into the appendices. An additional source of contamination resulted when a user needed an answer to a question and none of his attempts was successful. He might then telephone one of the system developers and ask

Table I. Number of events.

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	TOTAL
QUERIES	45	161	180	52	45	114	87	31	32	20	13	8	788
COMPLETED	18	92	127	44	28	66	68	22	17	17	9	5	513
LOOKUP FAILURES	3	19	17	5	10	19	31	8	1	2	3	1	119
PARSING FAILURES	14	26	38	4	4	29	11	7	11	0	3	0	147
NOTHING IN DATA BASE	12	8	12	3	1	2	13	5	1	1	3	0	61
ABORTED	3	9	14	2	5	9	4	2	1	2	1	1	53
USER COMMENTS	9	10	16	7	3	10	21	11	3	1	4	1	96
USER MESSAGES	0	3	0	0	0	3	4	1	0	0	0	0	11
OPERATOR MESSAGES	14	6	10	2	2	0	2	3	2	0	4	0	45
PROGRAM ERRORS	1	12	9	1	6	5	0	0	1	1	0	3	39
USER CANCELLED	1	5	1	1	2	5	4	0	2	0	0	0	21
LEXICAL CHOICE	6	32	11	4	10	31	6	3	1	11	3	1	119

Table II. Percentage of events to number of queries.

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	TOTAL
COMPLETED	40.0	57.1	70.6	84.6	62.2	57.9	78.2	71.0	53.1	85.0	69.2	62.5	65.1
LOOKUP FAILURES	6.7	11.8	9.4	9.6	22.2	16.7	35.6	25.8	3.1	10.0	23.1	12.5	15.1
PARSING FAILURES	31.1	16.1	21.1	7.7	8.9	25.4	12.6	22.6	34.4	0.0	23.1	0.0	18.7
NOTHING IN DATA BASE	26.7	5.0	6.7	5.8	2.2	1.8	14.9	16.1	3.1	5.0	23.1	0.0	7.7
ABORTED	6.7	5.6	7.8	3.8	11.1	7.9	4.6	6.5	3.1	10.0	7.7	12.5	6.7
USER COMMENTS	20.0	6.2	8.9	13.5	6.7	8.8	24.1	35.5	9.4	5.0	30.8	12.5	12.2
USER MESSAGES	0.0	1.9	0.0	0.0	0.0	2.6	4.6	3.2	0.0	0.0	0.0	0.0	1.4
OPERATOR MESSAGES	31.1	3.7	5.6	3.8	4.4	0.0	2.3	9.7	6.3	0.0	30.8	0.0	5.7
PROGRAM ERRORS	2.2	7.5	5.0	1.9	13.3	4.4	0.0	0.0	3.1	5.0	0.0	37.5	4.9
USER CANCELLED	2.2	3.1	0.6	1.9	4.4	4.4	4.6	0.0	6.3	0.0	0.0	0.0	2.7
LEXICAL CHOICE	13.3	19.9	6.1	7.7	22.2	27.2	6.9	9.7	3.1	55.0	23.1	12.5	15.1

for a suggestion. If that suggestion worked, this would inflate the percentage of success somewhat.

The TQA system incorporates two logging facilities. One of them is a verbatim record of all output that appeared on the user terminal. The other is a much more comprehensive trace of the system flow while it processed each question. The primary use of the second trace was to allow us to isolate the nature of the problems which arose with a view to correction. This file, however, contains much interesting detail about the amount of computer time, (as opposed to elapsed time), used in each processing step, the number of intermediate structures created, indications as to exactly which step caused a failure, and the like. Unfortunately, the second file was not originally meant to be amenable to machine processing, and its format was changed a few times during the course of the year. In addition, when the computing system failed, this second file was sometimes lost, although the basic log file seldom was. As a result, the statistics in the tables may be in error by a percentage point or two. In any case, the error is not sufficiently large to affect any of the major conclusions, which are qualitative at best.

Table I lists the raw numbers, by month and totaled for the year, for a number of the events which occur in system operation. Percentages for each event relative to the number of queries are given in Table II.

QUERIES refers to user inputs terminated by a question mark or exclamation point. COMPLETED is the number of queries which resulted in access to the data base and a resulting response to the user. LOOKUP FAILURES is the number of times the user typed a word which was not in the lexicon and was not capitalized. Such words were displayed back, with the option of changing the word entered or entering an entire new question. PARSING FAILURE means that the system did not reach the point of producing a query structure. NOTHING IN DATA BASE means that a logical form was generated and a query issued to System/R, but nothing satisfied the query. This could happen for a variety of reasons, including an erroneous parse, a wrong logical form, a mistake in the program which generated the search request, or a real case of missing data. Detailed analysis of each case would be required to be sure what proportion fell into each category.

The other categories are more or less self-explanatory. ABORTED is an indication that query

processing did not reach a normal termination, usually because of a machine failure, i.e., hardware or system software, but sometimes because of a problem in the TQA code. The USER COMMENTS line comes from a feature we had included in the system in an attempt to collect on-going user reaction to system behavior. At the end of each question, before producing the message "TYPE NEXT QUESTION", we displayed a request for comment on the preceding question. It turned out that users found this something of a nuisance, since mostly they were satisfied with the answer, so we made an early change to the terminal read program such that a reply to the prompting message for comments which ended in "?" would be treated as a null comment and another question, in effect making the comment optional. Student users took the prompt somewhat more seriously than the regular employees and made more comments. The two MESSAGE categories refer to a system facility enabling a user to send messages to the TQA operator and the reverse. One can see that it is usually used by the TQA operator as a somewhat more convenient means than the telephone for advising a user about the nature of a problem or for warning that the system was to be stopped, and the like. PROGRAM ERROR is an error in the TQA program which was detected and caused a query to abort but left the system able to accept another input. There is also a facility, tabulated under USER CANCELLED, allowing a user to cancel a query at any time by pressing a special key on the terminal keyboard. This is used when the user realizes he made a mistake or when processing seems to be going on too long. The category LEXICAL CHOICE indicates how often the system asked the user to clarify the meaning of a word. For example, "area" can be "parcel area" or "floor area" and if the system cannot disambiguate by the context, the user is shown the two choices and asked to "Type A or B". The most frequent ambiguities have to do with duplication of names for streets, neighborhoods and schools.

As can be seen in line 1 of Table I, there was a drastic fall-off in usage in the latter part of the year. This has two primary causes. In the first place, the planning office was being rebuilt during the period and the work space was often disrupted. The major cause, as mentioned above, was a re-orientation of the planning department activities by a newly appointed director. The department members now spend the major part of their time on administrative activities; planning activities, like land analyses and drafting of new zoning districts, are now carried out by off-premises consultants who do not have ready access to the terminal. From the latter part of 1978 through 1979, the system was used only intermittently by the planners, who occasionally needed some of the basic land record data, and by others, like the fire department, who wanted to have a list of tall buildings.

There is an obvious rise in USER COMMENTS, LOOKUP FAILURES, and PARSING FAILURES during June, July, and August. During this period, the planning department had a number of student interns who were using the system, sometimes in a play mode. It is easy to understand that new users exploring the system would have more than an ordinary number of parsing failures. The increase in lookup failures is a little more puzzling, although part of the increase has the same cause, i.e., some of the questions were outside the domain, and consequently the words used were not in the lexicon.

The large percentage for the NOTHING IN DATA BASE category in January comes in large part from two queries which had four underlying structures, all of which resulted in the answer "NOTHING IN THE DATA BASE." This was caused by a problem in the grammar which has since been corrected. Some of the other instances of this message during the year resulted from an inadvertently, and unfortunately, successful example of subtle user training or conditioning. We discovered that users tended to respond to the system by echoing, sometimes exactly and sometimes only partially, what the system had printed to them, no matter if their initial phrasing had been accepted or not. Some input word sequences are treated as phrases by the strategy of scanning an input query against a phrase lexicon first before looking in the regular lexicon. An entry like "gas station" came out of the phrase lookup represented as "GAS_STATION" for purposes of lexical lookup. The users would sometimes input that, or some variant like "Gas Station", which would be taken as a proper name. The result would then be interpreted as a query having to do with an owner named "Gas Station". We now echo what the user has typed, or some variant which will be acceptable if typed. Most of the instances of the category NOTHING IN DATA BASE really are the response to a request for information which is not in the data base. Many of these are requests for information about people who are not owners of property, or about addresses which are not legitimate.

Apart from the drastic fall in usage at the end of the year discussed above, there appears not to be any trend in any of the other measures of system operation. The sequence of peaks and valleys may be characteristic of systems like this, or may simply result from insufficient time for the system to settle down.

Operating Characteristics

Cumulative statistics for the system operating characteristics are found in Tables III and IV and Figures 4 through 8 for each system component. The histograms have a Poisson-like shape, with a single peak

Table III. Query/logical form time - No. of structures.

	1	2	3	4	5 >
TOTAL YEAR					
QUERY STRUCTURE TIME (SEC)	497	53	22	16	3
LOGICAL FORM TIME (SEC)	536	24	4	2	12
NO. OF SURFACE STRUCTURES	470	146	38	9	3
NO. OF UNDERLYING STRUCTURES	581	7	0	2	0
NO. OF QUERY STRUCTURES	581	7	0	2	0
NO. OF LOGICAL FORMS	571	5	0	2	0
NO. OF ANSWERS	513	0	0	0	0

Table IV. Query/logical form time - No. of structures by month.

	1	2	3	4	5 >
MAY					
QUERY STRUCTURE TIME (SEC)	26	3	7	0	0
LOGICAL FORM TIME (SEC)	29	5	2	0	0
NO. OF SURFACE STRUCTURES	6	26	5	3	0
NO. OF UNDERLYING STRUCTURES	36	0	0	0	0
NO. OF QUERY STRUCTURES	36	0	0	0	0
NO. OF LOGICAL FORMS	36	0	0	0	0
NO. OF ANSWERS	28	0	0	0	0
JUNE					
QUERY STRUCTURE TIME (SEC)	35	18	10	7	2
LOGICAL FORM TIME (SEC)	64	4	2	2	0
NO. OF SURFACE STRUCTURES	34	31	21	3	0
NO. OF UNDERLYING STRUCTURES	72	0	0	0	0
NO. OF QUERY STRUCTURES	72	0	0	0	0
NO. OF LOGICAL FORMS	72	0	0	0	0
NO. OF ANSWERS	66	0	0	0	0
JULY					
QUERY STRUCTURE TIME (SEC)	58	10	1	1	0
LOGICAL FORM TIME (SEC)	69	0	0	0	0
NO. OF SURFACE STRUCTURES	62	12	1	0	0
NO. OF UNDERLYING STRUCTURES	70	0	0	0	0
NO. OF QUERY STRUCTURES	70	0	0	0	0
NO. OF LOGICAL FORMS	69	0	0	0	0
NO. OF ANSWERS	68	0	0	0	0

and a long tail. (The apparent spike at the right-hand end of the "Number of lines in answer" is not real; that point is really "answers of 50 lines or more". The right-most point of all the histograms should be read in a similar way, i.e., as including the total of all values in the remainder of the tail.)

Tables III and IV contain two kinds of information. The first two lines under "Total Year" in Table III and under each month in Table IV show the number of queries whose machine processing time required the number of seconds shown by the column head. Thus, in Table III, 4 queries took 3 seconds of processing time to generate a logical form. A table form was used for this information because a histogram for "time to produce a query structure" and "time to produce a logical form" would have been simply a spike. The other lines in the tables are counts, so that,

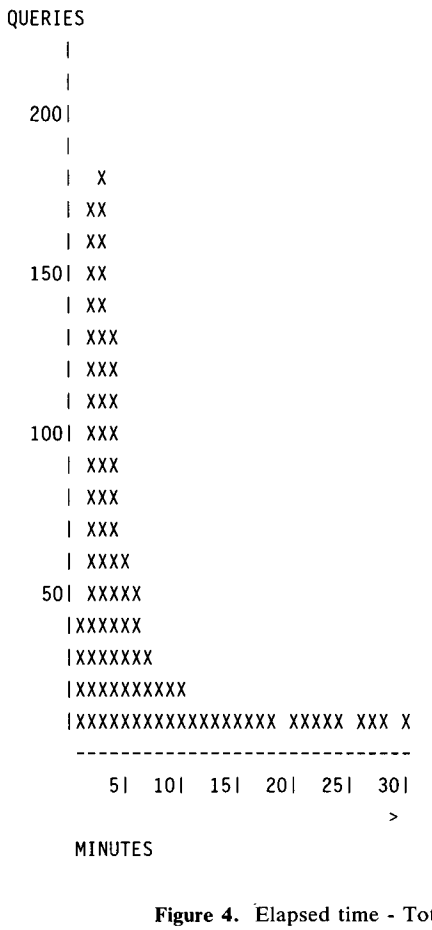

Figure 4. Elapsed time - Total.

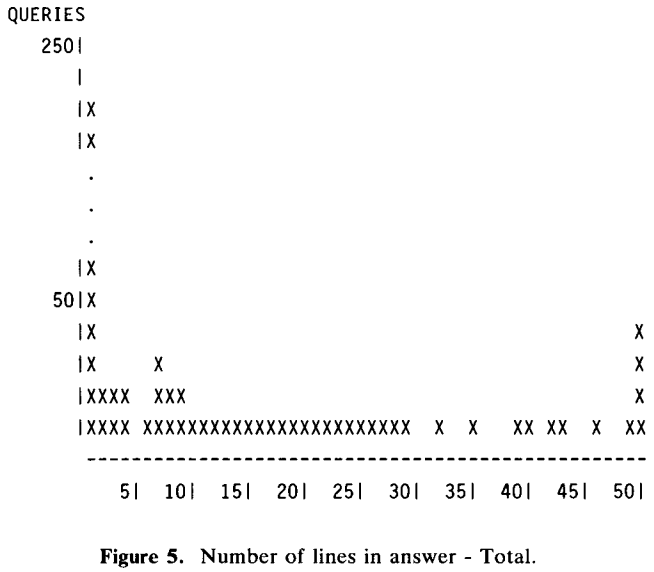

Figure 5. Number of lines in answer - Total.

again in Table III, 7 queries had 2 underlying structures.

From the first two lines of Table III, it can be seen that neither the query structure nor the logical form account for very much of the processing time, mostly

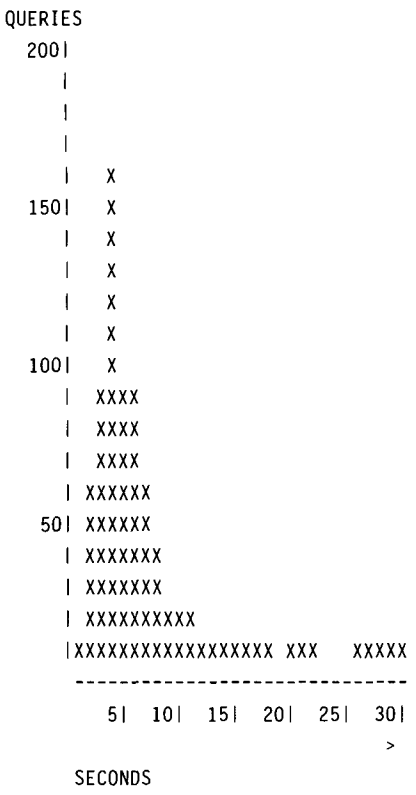

Figure 6. Generation of surface structures - Total.

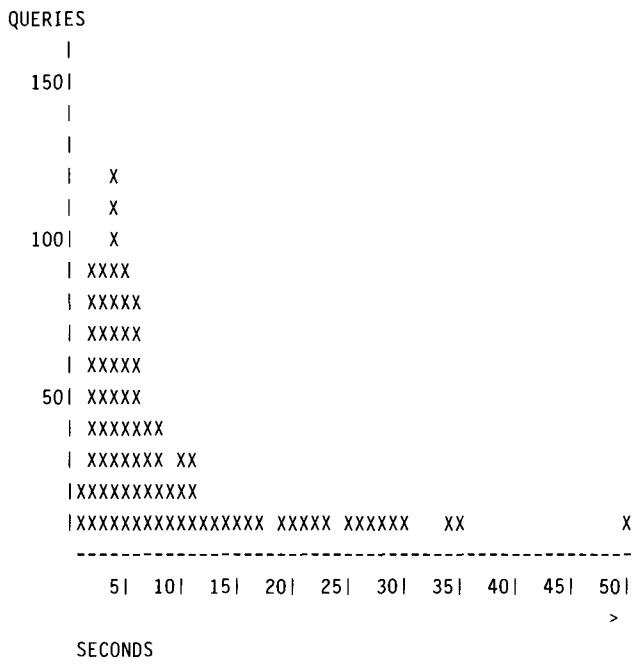

Figure 7. Transformational parsing - Total.

less than 1 second, and Figure 8 shows that answer processing, i.e., data base lookup and printing, is also not a major time user. From Figures 6 and 7 it can be seen that surface structure parsing, which includes the

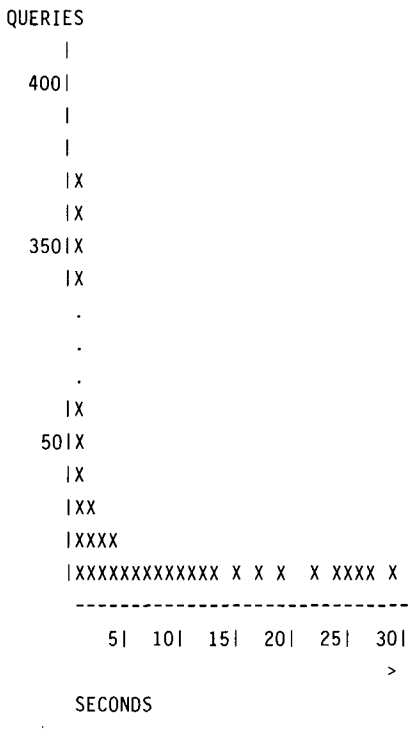

Figure 8. Answer processing - Total.

time for string transformations, and transformational processing both typically consume around 4 seconds. Therefore, total machine time for a typical query is around 10 seconds, although extreme cases can take much longer.

From Figure 4, it can be seen that elapsed time for a query is around 3 minutes, although there is again a long tail. Elapsed time depends primarily on machine load and user behavior at the terminal. The computer on which the system operated was an IBM 370/168 with an attached processor, 8 megabytes of memory and extensive peripheral storage, operating under the VM/370 time-sharing system. During business hours, there were usually in excess of 200 users logged on to the system, so any one user received only a small share of the resources. Besides queuing for the CPU and memory, this system developed queues on an IBM 3850 Mass Storage System, on which the TQA data base was stored. With all delays accounted for, it could easily take several minutes of elapsed time to accumulate 10 or 15 seconds of CPU time.

User-caused delays tend to occur when a reply is needed to correct a lookup failure or to resolve an ambiguity, and when the answer to a query requires more than one screen for display. In the latter case, the display has a built in delay of up to one minute, or can be held indefinitely by depressing a key. The hold feature is often used for the long displays, because the user is writing down information from the display.

The user is, of course, not concerned with his own delays, but only with the system delay. In order to keep him informed of progress through the query, and give him assurance that the computer is still operating, a CPU time clock is displayed in the lower right corner of the screen, along with an indication of the processing phase being carried out at that time. In general, these users did not become concerned with response time, possibly because they had no experience with other interactive computer systems. However, if the system was slow, a user might well choose to look up a single piece of information in the printed listing of the data base rather than ask through the computer system.

Figure 5 shows that most queries have a one line answer, but that the tail is very long. Generally speaking, users were interested in totals or averages of data fields like "dwelling units" or "parcel area" for aggregates of parcels specified by land use, neighborhood, census tract, and the like, often in combination. Questions of this kind are not readily answered from the printed copies of the parcel file, and specific patterns do not occur often enough to make it worthwhile to ask for a batch program to be written by the City computer staff.

The yearly summary statistics discussed above conceal the considerable variation which can be encountered over shorter time periods. Table IV shows the monthly statistics, corresponding to Table III for May, June and the more typical July. The percentage of queries with more than one surface structure is clearly very high in May and June. This results in longer average times for query structure processing and logical form processing, as seen also in Table IV. The effect is even more obvious in the histograms for surface structure processing, Figures 9-11, and underlying structure processing, Figures 12-14. Inspection of the logs for May and June shows that most of this effect results from only a few questions, e.g., "What is the area of the *x* family houses in the *y* zone?", repeated for a number of combinations of x and y. It happens that one of the planners was checking figures to be included in a table in a large report during this period. Sequences of questions like this are, of course, of little help in system development, but do serve to build confidence in the utility of the system.

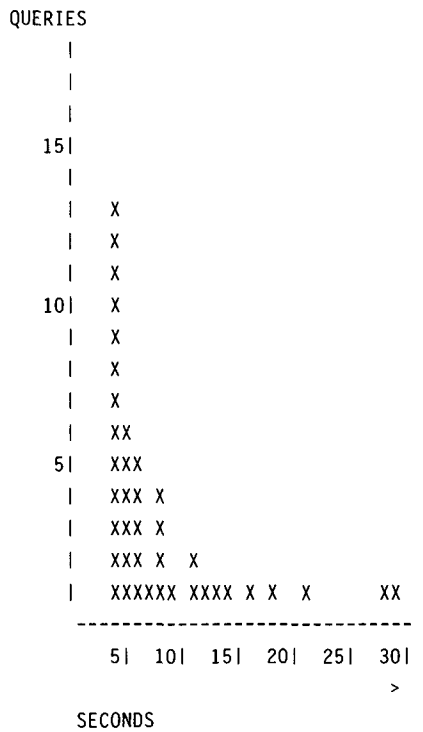

Figure 9. Generation of surface structures - May.

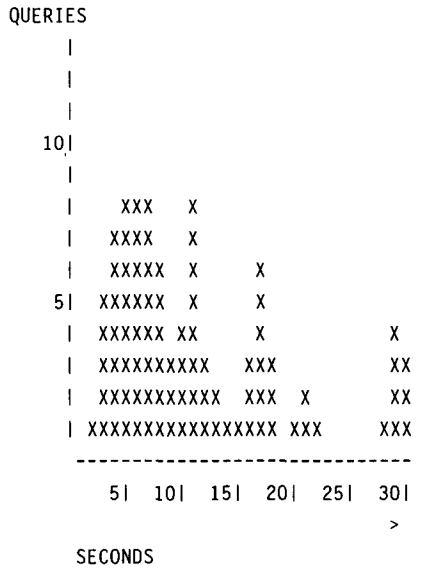

Figure 10. Generation of surface structures - June.

The Input Queries

Figure 1 given earlier shows a small selection of the queries put to the system during 1978, to give the reader an idea of the kinds of questions asked by the users. Additionally, there are four lengthy appendices *which appear only in the microfiche supplement to this issue of the Journal.* Appendix A presents the entire set of queries submitted to the system during 1978, (but note the caveat at the beginning regarding possible missing data because of lost logs). Sentences preceded by a *C* in Appendix A were completed successfully, but one must remember that "NOTHING IN THE DATA BASE" was counted as a successful answer by the data reduction program even though there may have been some earlier problem in the query. Appendix B is a list of sentences which did not parse

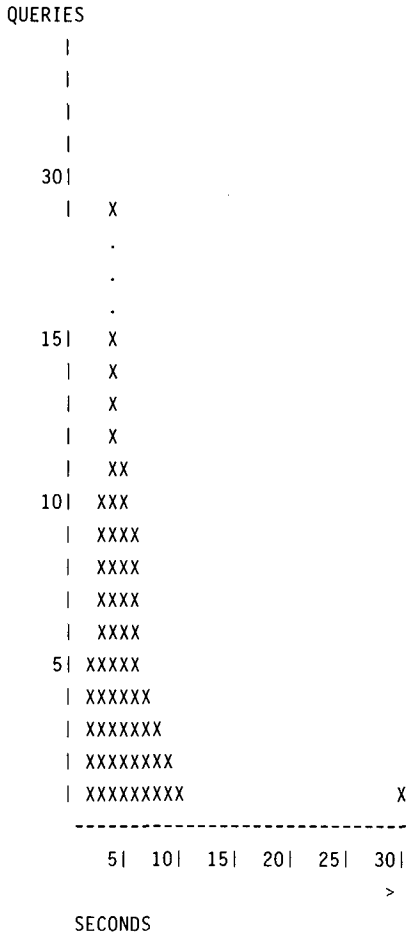

Figure 11. Generation of surface structures - July.

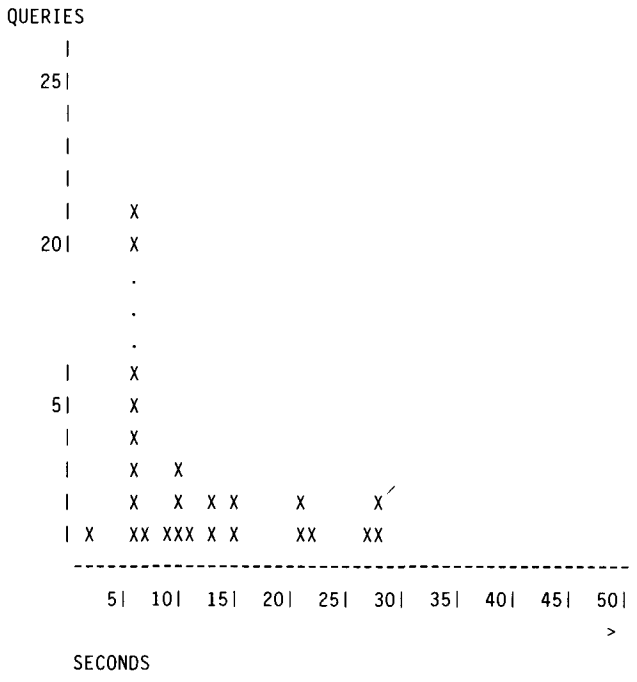

Figure 12. Transformational parsing - May.

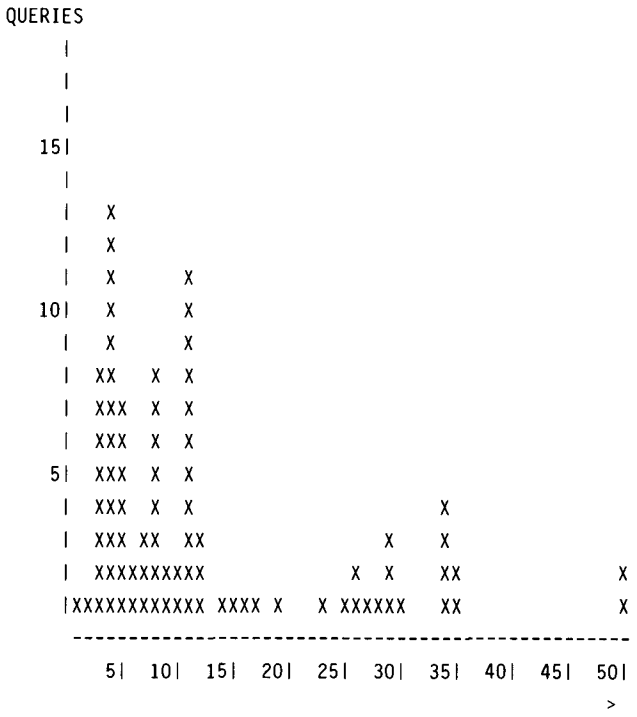

Figure 13. Transformational parsing - June.

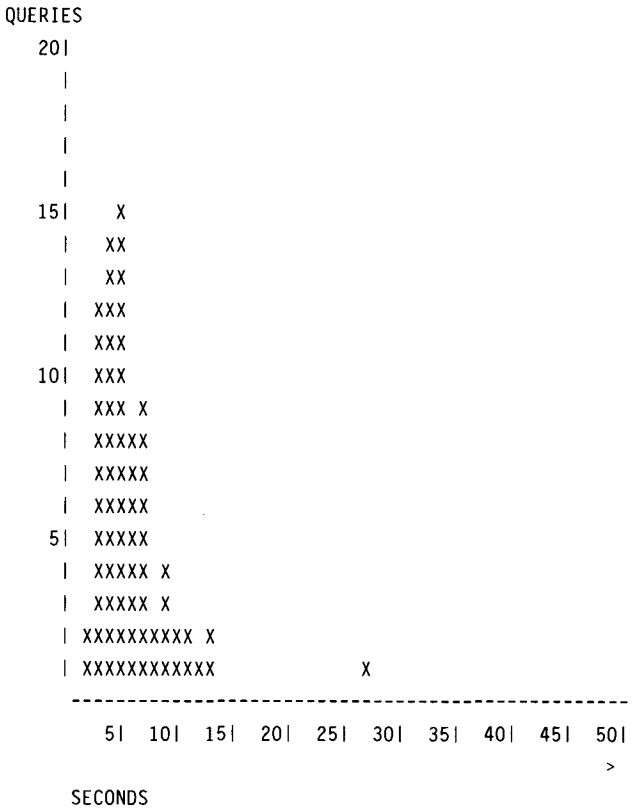

Figure 14. Transformational parsing - July.

at their first submission, and Appendix C is a list which failed for some reason other than parsing at their first submission. Appendix D is a list of sentences for which the answer was "NOTHING IN THE DATA BASE." In most cases where this answer was not correct, the problem was in dealing with searches on persons' names. The treatment used presently is still not completely satisfactory, but has been improved since the initial version. Names in the data base occur in a great variety of patterns, more than we felt worth devising procedures to handle. In the three latter appendices, those sentences which are satisfactorily answered by the system of May 14, 1979 are preceded by an *X*. (Any necessary spelling corrections or ambiguity resolution were supplied.) A little over 60 percent of the failing sentences are processed correctly by this version. As can be seen, many of the remaining queries are so flawed that no system should be expected to answer them.

The set of successful queries is not really indicative of the full coverage of the system. There are a number of permissible constructions which the planners simply never used, so the subset is somewhat richer than shown. On the other hand there are large numbers of constructions involving personal pronouns, three-argument comparatives, quantification with "each", etc., which the planners knew the system did not handle, and which they consequently did not use.

Conclusions

The motivation for publishing this paper with its long appendices was to make it possible for a reader to come to some conclusion of his own regarding the utility of the TQA system or something like it in a real environment. For reasons mentioned above, none of us thinks this experiment provides a definitive test for our system or even that very strong claims about percent of success can be made. For such purposes, a controlled experiment using a fixed system is clearly necessary, and we hope to make such a test in the near future. However, those of us working on the project are encouraged by the results summarized here, and even more by our conversations with the users of the system, who have been very positive. Within the limitations established by the subset of English that the system can recognize, it appears that non-data processing professionals can extract useful information from their files with almost no training. A number of the gaps in our system can certainly be plugged if we can find the resources to work on them.

Our next step will be to make the TQA system a front end to an existing formal query language system, probably SQL-based System R, developed at the IBM San Jose Research Laboratory, a version of which has been announced as an IBM Program Product for use under the DOS/VS operating system. Such a step will permit us to devote all of our resources to the problems of language and knowledge representation, instead of spending part of that effort on data management. Beyond that, we will consider seriously the question of moving to different environments, both to new situations in the city planning domain and to completely different domains, without having to rewrite all or even a major portion of our base system.

Acknowledgements

Besides Petrick, Plath and the author, the TQA project currently includes Mark Pivovonsky, who has done the systems programming for the project. Since this document summarizes one year's activity, it necessarily reflects the content of many discussions with these individuals, and some of their opinions and insights on the nature of these problems are incorporated herein. The wrong ones naturally are mine.

References

Astrahan, M.M.; Blasgen, M.W.; Chamberlin, D.D.; Eswaran, K.P.; Gray, J.N.; Griffiths, P.P.; King, W.F.; Lorie, R.A.; McJones, J.; Mehl, J.W.; Putzolu, G.R.; Traiger, I.L.; Wade, B.W.; Watson, V. (1976). System R: Relational Approach to Database Management. *ACM Transactions on Database Systems,* 1 (21), pp. 97-137.

Damerau, Fred J. (1977). Advantages of a Transformational Grammar for Question Answering. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* vol 1, p. 192.

Damerau, Fred J. (1978). The Derivation of Answers from Logical Forms in a Question Answering System. *American Journal of Computational Linguistics,* Microfiche 75, pp. 3-42.

Krause, Juergen. (1979). Results of a User Study with the 'User Specialty Languages' System and Consequences for the Architecture of Natural Language Interfaces. Technical Report 79.04.003, IBM Heidelberg Scientific Center, May 1979.

Petrick, Stanley R. (1977). Semantic Interpretation in the Request System. In Computational and Mathematical Linguistics, *Proceedings of the International Conference on Computational Linguistics,* Pisa, pp. 585-610.

Plath, Warren J. (1973). Transformational Grammar and Transformational Parsing in the REQUEST System. *IBM Research Report RC 4396,* Thomas J. Watson Research Center, Yorktown Heights, N.Y.

Plath, Warren J. (1974). String Transformations in the REQUEST System. *American Journal of Computational Linguistics,* Microfiche 8.

Plath, Warren J. (1976). REQUEST: A Natural Language Question-Answering System. *IBM Journal of Research and Development,* 20:4, (July 1976), pp. 326-335.

Robinson, Jane J. (1973). An Inverse Transformational Lexicon. In *Natural Language Processing,* Randall Rustin, ed., Algorithmics Press, Inc., New York, pp. 43-60.

Fred J. Damerau is a Research Staff Member at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. He received the Ph.D. degree in linguistics from Yale University in 1966.