II. Operating Instructions for the SMART Text Processing and Document Retrieval System

M. E. Lesk

1. Introduction

SMART is a fully automatic programming system designed for the analysis of English text and for the processing of information requests. The SMART system can also be used to perform complete studies of information retrieval systems, vocabulary and dictionary investigations, and many other linguistic and classificatory studies. Among the features of the system which are available for purposes of content analysis are: thesaurus lookup, phrase searching methods (including complete syntactic analysis), hierarchical processing, statistical association methods, and title/abstract/text differentiation. Intermediate and final results are available in convenient form for detailed study, or alternatively, a complete retrieval technique may be evaluated automatically. A full description of the methods and programs used may be found in [1]; shorter writeups are available in [2], [3], and [4].

The system described here supersedes the operating system described a year and a half ago in [5] and [6]. The basic differences between new and old systems are a widely expanded system capacity and faster processing speeds. In future years, it is hoped to experiment with time-sharing environments, and additional processing algorithms are being developed. Since the programs have reached a state of semi-stability, however, new operating instructions are being prepared.

It is assumed that the reader has a general understanding of the purpose of the SMART system. Reference to [2] or [5] provide this background. The main purpose of the present section is to replace the obsolete operating instructions in [6].

1.1. Processing Summary

SMART accepts as input natural language English text in a form close to normal typing. Input is basically of two forms: requests, or queries for information; and documents, or individual units of a collection of English articles, or abstracts, which are compared with the requests.

The search procedure makes use of a flexible data representation system in which each document is analyzed into a "concept vector". The concept vector consists of a list of "concepts" (each with a weight) that are associated with the document. If each possible concept is imagined as a dimension, or a direction in a very-high dimensional space, the "concept vector" representation of a document is now equivalent to an n-dimensional vector; a concept vector is thus readily compared with other, similar representations by simple correlation procedures.

The meaning of a "concept" can differ greatly from run to run. A concept may be simply an English work; or it may be a set of English words; or it may be a phrase or set of phrases; or it may be a node label in a hierarchical classification system; or any combination of the above. Provision is also made for future inclusion of other types of concepts; e.g., concepts derived from an author's name or from a journal citation. In short, a concept may represent virtually any feature of the text of a document which reflects the document content.

The analysis of English language text for the detection of concepts is performed without human intervention. Text is analyzed by a set of programs which operate in conjunction with two types of inputs:

a) A set of dictionaries, grammars, and hierarchies specifying the relations between properties of the input English text and of the concepts. The simplest possible dictionary, for example, associates with each English word a distinct concept. A more sophisticated dictionary, consisting in fact of a word thesaurus, will define a many-to-many mapping of English word stems into the concept vector space in which some words are isolated as common words, some are considered ambiguous and resolved into their various possible meanings, and some are combined with synonymous terms.

b) A set of specifications which describe in what way the various content analysis programs are to be applied and which dictionaries are to be used. Specifications are used by the program to decide what documents are to be processed, what dictionaries should be used, what algorithms for associating concepts with documents should be employed, how much weight each contribution to the document representation should be given, what procedures should be utilized to compare requests with documents, and what output is desired from the run.

Once the text analysis has been performed, and concept vectors are available for all documents and all requests that have entered into the system, the programs proceed by comparing the requests with the documents to determine which documents are to be identified as "answers" to the requests. This list may then be compared with a "correct" list, obtained as a result of a manual operation and supplied by the programmer, and various evaluation measures may be computed automatically.

1.2. Operating Programs

The programs available fall into three categories:

a) Retrieval system programs consisting of a large, connected set of programs which perform all the steps described in part 1.1. These programs use a set of dictionaries to operate on a collection of English text, and produce answers to requests, and evaluations of processing methods.

b) Dictionary preparation programs consisting of a smaller, but also connected set of programs which prepare the various tapes required by the program (a).

c) Miscellaneous programs to perform various useful functions outside the main system.

A description of the instructions for running programs (a) is given in parts 2, 3, and 4 of these instructions. Programs (b) can be run using the instructions in part 5, and programs (c) are discussed in part 6.

2. Basic Operating Procedures

SMART operates under a normal FORTRAN II monitor system. As already mentioned, the programs require a set of instructions, a library tape, and a set of data. The usual procedure is to supply the programs on one tape, the library on a second tape, and the data on another private tape and/or on the monitor input tape. In addition, up to seven scratch tapes may be needed. Any data for the monitor input tape follows the run specifications.

2.1. Run Outline

A complete processing run proceeds as follows:

a) Monitor control cards are supplied to sign on to the computer,

to instruct the operator to mount necessary tapes, and to transfer control to a small program called CLCHN which calls in the SMART programs from the program tape.

b) The instructions for the run are read from the monitor input tape and decoded.

c) The English language input text is read and analyzed. This includes a lookup of the text in a thesaurus, and an optional lookup in a statistical phrase dictionary and a syntactic phrase dictionary. Printouts of the original text and the words missing from the dictionaries may also be performed.

d) Partial concept vectors are formed for these documents. These partial vectors contain all concepts derived from individual words and phrases within each document. The concepts appear in their original uncombined form.

e) Partial concept vectors from documents looked up in this run, and concept vectors from documents looked up in preceding runs are collated, and the weighting schemes specified in the instructions are applied to produce concept vectors in the proper form for correlation. If desired, the detailed results of the weighting may be printed.

f) The concept vectors may now be expanded by means of statistical concept-concept associations, or through a hierarchy associated with the thesaurus. Various options may control the type of expansion, and may specify the documents to be expanded.

g) The request vectors are compared with the document vectors using the correlation procedure chosen by the user. All documents in the collection are ranked with respect to each request.

h) The highest ranking documents are printed out for each request. This is the basic output of the simulated retrieval system.

i) The document rank list and the highest ranking documents are printed and compared with the list of documents previously designated as relevant to the request. Measures are computed which

indicate the success or failure of the run, relative to other runs.

The instructions to be used to regulate these procedures are described in part 3; the data needed by the programs are discussed in part 4.

2.2. Tape Setup

The tapes used by SMART are named AONE, ATWO, ATHREE, ... AEIGHT, BONE, BTWO, BTHREE, ..., BSIX. By means of two internal tables, these names are translated first into logical tape numbers and then into physical tape unit addresses. This last translation takes place using the operating system table (IOU), and SMART is therefore adaptable to almost any tape assignment of the individual monitor system. To aid in running SMART under a given monitor, the first table (which translates SMART tape names to logical numbers) is variable at object time through the normal specification system. Unless altered in this way, the SMART tapes correspond to the 7094 tape drives with the same names under the (IOU) table of the Harvard monitor (see Table 1). Because of this convention, the distinction between SMART tape AFIVE (for example) and the physical drive A5 will not be made in this writeup, although the correspondence is indirect.

SMART assumes that input will be on A2 (actually ATWO) in accordance with the practice in use for most FMS operating systems. Output print is placed on A3; punching is done through the offline punch tape B4. The SMART program tape is placed on B3; it is a normal FMS chain tape (see part 5.3). The library tape is mounted on unit B5; any input data tape goes on A6. All remaining tapes (A4, A5, A7, A8, B1, B2, B6) are used as scratch tapes. A4 is needed for any lookup, and is also used if ANSWER MEDIUM or ANSWER LONG is requested. A5 is needed for the lookup, and for

any correlations or expansions. A7 is needed for correlations or expansions, or for syntactic processing. A8 is needed only for concept-concept or hierarchical expansions. B1 is needed for expansions and for syntactic processing. B2 is needed for correlations, syntactic processing, and expansions. B6 is needed for correlations, phrase lookup, and expansions. If a tape is not needed by the specified process, it need not be mounted.

2.3. Input Deck Setup

Since the SMART system runs under a normal FORTRAN II monitor system (FMS), the usual deck setup is employed. The job card (system accounting data) appears first; then come operator messages and monitor control cards. The program, consisting usually of a short routine which merely calls in the main programs from the program tape, follows. After this short loading deck (labeled CLCHN) comes the monitor * DATA card. The data which follow consist of the instruction cards and the data for the processing runs. The instructions are on A2. The data, which usually consist of requests, texts, and relevance judgments, follow and may be on either A2 or A6. Several processing runs may be stacked in the same FMS job, as explained in the next section. Alternatively, each processing run may be made a separate FMS job. This is often preferable if new tapes must be mounted, since the SMART system makes no provisions for pausing to allow the operators sufficient time to change the tapes.

3. Specifications for the SMART Retrieval System

The first date required by the SMART programs are the specifications which direct the processing to be performed during the computer run. These specifications must immediately follow the monitor * DATA card.

Specification input always appears on the monitor input tape (FORTRAN logical tape 5), even if SMART tape ATWO is reassigned to another unit. This exception to the tape addressing system is made to permit the program to get started. One computer job may involve several sets of specifications, each of which initiates a processing sequence. In this case, each set of specifications is followed by the data required by that set of instructions. The AGAIN specification (see part 3.8) causes the system to expect a second set of instructions and data following the first set.

Each section of a SMART processing run begins with the specifications for that section. The specifications are punched in columns 1-72 of as many cards as are needed. A specification consists of an alphabetic word which identifies the specification, followed by its "value" if it is not an "on/off" specification. The names and meanings of all specifications, and their allowed values, are given in part 3. Whenever a specification differs from the normal value for the item specified (see Table 2), the user must punch the specification and the value (if any) on the specification cards. Specifications which are given their normal value may be omitted, or may be punched to improve the clarity of the listing, if the user so desires.

Any specification may be punched anywhere on a card. However, no specification may be broken between two cards, nor may a specification and its value appear on different cards. The specifications are separated by one or more blanks; the value of a specification should follow the specification itself separated by one or more blanks. If the value of a specification is a number, care must be taken to punch it correctly. If the value of a specification is restricted to integers,

the number given must not have a decimal point (2, not 2.0). If the value of a specification is allowed to be a fraction, it must be punched as a FORTRAN floating-point number, i.e. it must have a decimal point (or the letter E). For example, STATWT 1 is incorrect; the specification must read STATWT 1. or STATWT 1.0. In Table 2, x indicates a floating point number and n indicates an integer.

If a specification appears more than once, only the last occurrence is used. Specifications may occur in any order, except that the last specification (and only the last) must be either STOP or X.

3.1. Specifications Affecting Lookup

The SMART lookup uses a stem dictionary together with a suffix list for an accurate determination of both semantic and syntacic roles. The dictionary is stored in a semi-tree format, as is the suffix list. Details of the lookup may be found in references [11] and [12].

It should be noted that the lookup is sufficiently accurate so that if, for example, HOPE and HOP are two stems in the dictionary, HOPPING will be found as HOP + P + ING while HOPING will be found as HOPE - E + ING; also, if EASY and EASE are stems, EASIER is found from EASY while EASING is found from EASE.

The specifications associated with the lookup are:

ENGTXT causes the English text being looked up to be printed;

NOTFND causes the words not found in the dictionary to be printed;

PUNCH causes document vectors to be punched out (with phrases, if searched for) in binary form. This saves time in future runs since the lookup need not be repeated.

3.2. Specifications Affecting Phrase Searching

Two basic phrase searching methods are available. These are called the statistical and syntactic phrase searching procedures. Both use pre-designed dictionaries of phrases which are searched against sentences in the texts. Statistical phrase searching is the more common of these methods. The phrase dictionary consists of a set of pre-assigned word pairs or word n-tuples (where n can be 2,3,4,5 or 6). Each word n-tuple has a "phrase concept number" associated with it, that is used as the concept number representing the whole phrase (as distinguished from the individual concept numbers which give the components of the phrase). Every sentence is scanned for occurrences of phrase components, and when all of the n phrase components have been found to occur at least m times in the sentence, the phrase is considered to have occurred m times, and a weight is entered accordingly for the "phrase concept number". The method of writing the phrase dictionary on the library tape is given in part 5.1.2.

It is noted that this is a rather imprecise type of search, since the words searched for may occur anywhere in the sentence in any order. For example, the sentence "Despite a second, larger order of textbooks, approximately twenty-five percent of the students are still without them" will be considered as containing the phrase "second order approximation". A more precise type of search is available through the "syntactic phrase searching" procedure. This uses the "criterion tree" dictionary (see 5.4.4) in which a complete syntactic/semantic/structural specification of each phrase is given. If a syntactic phrase is to be detected,

1) the components of the phrase must have the proper semantic

value, i.e. they must be assigned the proper word stem or the proper thesaurus category;

2) the components of the phrase must have the proper syntactic role (e.g. "automatically translating" would not match "automatic translation");

3) the components of the phrase must exhibit the proper syntactic dependency relations with each other (e.g. "blind Venetian" would not match "Venetian blind").

This phrase searching procedure makes use of the Kuno multiple-path syntactic analyzer [7,8,9] to determine the syntactic roles and dependency relations of the words in the sentence. This makes possible very accurate searching, in that, for example the phrase "solid state" would not be recognized as appearing in "Some authors whose other work is solid state that information retrieval can be performed without accurate syntactic analysis."

It should be noted that the syntactic phrase searcher is very slow and also very erratic in its timing. It is hoped that the use of a new version of the Kuno syntactic analyzer will alleviate these problems [10].

The specifications to be used in the phrase searching procedure are:

STATPR which causes a statistical phrase search to be performed;

SYNTAX which causes a syntactic phrase search to be performed;

NODECO causes a table of the detailed correspondences between tree nodes and sentence words to be printed;

SYNANA causes the syntactic analyses of the text searched to be printed.

An additional phrase searching method can be envisioned in which statistical properties of words are analyzed to determine which words appear to be occurring as phrases. SMART has in the past contained such procedures, but due to the poor results obtained with the algorithms tried, these have not been implemented in the present version. New algorithms are under consideration, however, and specification CLUSPR has been maintained in the supervisor, indicating a call to a lookup routine using statistically detected phrases. However, this specification is currently inoperative.

3.3. Vector Expansions by Means of Concept-Concept Correlation

Concept vectors obtained from the words and phrases in one document may be expanded based on statistical data obtained from an entire document collection. In this way, local variations in individual document vocabularies can be corrected. Procedures for these statistical expansions are available in SMART through the form of concept-concept correlation. This option involves the formation of a complete concept-document occurrence matrix from the concept vectors of all the text submitted in the run. Different concepts are now correlated, to find out which concepts appear to exhibit a similar occurrence pattern in the documents. This comparison is made on the basis of the rows of the transposed concept document matrix in which each concept is represented by rows of elements consisting of the numeric occurrences of the concepts in the successive documents.

The correlation algorithms are as follows. Let the concept-document matrix be called $\underline{A}$, so that A_{ij} represents the number of occurrences of concept $\underline{i}$ in document $\underline{j}$. Then, the correlation coefficient r_{pq} is a

measure of relation between the rows A_{pi} (all i) and A_{qi} (all i). It has the value 0 for totally dissimilar rows, and 1 for identical rows. Two different numerical algorithms are available for the computation of $\underline{r}$; these are the cosine algorithm and the overlap algorithm. The cosine algorithm is defined as follows:

$$r_{pq} = \sum_k (A_{pk} * A_{qk}) \; / \; (\sum_k (A_{pk}^2)) * (\sum_k (A_{qk}^2)).$$

The overlap algorithm is defined as follows:

$$r_{pq} = \sum_k (\min(A_{pk}, A_{qk})) / (\min(\sum_k A_{pk}, \sum_k A_{qk}))$$

where min(x,y) = the numerically smaller of x and y. Note that both measures are symmetric; i.e., $r_{pq} = r_{qp}$.

These correlations can now be subjected to a cutoff process defining $\underline{s}_{ij}$ as follows: $s_{pq} = 1$ if r_{pq} is greater than the cutoff value; $s_{pq} = 0$ otherwise. The matrix $\underline{S}$ with both rows and columns labeled by concepts now identifies the concepts which have similar document environments; that is, the pairs of concepts which occur in the same documents have a one at the intersection of their row and column.

This process can now be used for the expansion of the document vector by augmenting all concepts by the list of concepts with similar environments.

The similarity matrix $\underline{S}$ can also be used as the starting point for further correlations, however. Some writers feel that the truly significant question is not "which concepts have similar document environments" but "which concepts have similar concept environments". This question requires an additional correlation, a correlation of the matrix $\underline{S}$, to identify the

concepts which co-occur with similar concepts. For example, if one is seeking to identify that "aeroplane" and "aircraft" represent the same concept, one might deduce this not from the fact that they co-occur in the same document, but that they each separately co-occur with words such as "fusilage", "propeller", "aileron", and the like. The SMART system makes provision for iterated concept-concept correlation to any depth.

The specifications affecting this procedure are as follows:

CONCON n — the number "n" is the number of iterations of the concept-concept correlation process. "CONCON 1" specifies simple first-order correlation;

MODECC a — a is either "COS" or "OVLAP" to specify cosine or overlap correlation for the first concept-concept correlation;

MODE2C a — a is either COS or OVLAP to specify the correlation mode for concept-concept correlations after the first correlation, i.e., for the second and following iterations;

CUTCC x — x is a number between 0 and 1 (0.6 is typical) to specify the cutoff for the first concept-concept correlation;

CUT2C x — x is a number between 0 and 1 specifying the cutoff for correlations after the first;

CONMIN n — n specifies the lowest concept number which will be correlated. This specification is to be used primarily with null dictionaries prepared by THES (see part 6.1) in which the concepts are arranged by frequency, thus making it possible to specify the lowest frequency word to be correlated;

CONMAX n — n specifies the highest concept number which will be correlated. CONMIN and CONMAX are used because the statistical procedures are not accurate for words which occur either very rarely or very frequently;

EXPAND a — a is either REQS, DOCS, or ALL and specifies whether requests only, documents only, or all input is to be expanded.

3.4. Vector Expansion by Means of Concept Hierarchies

In addition to statistical means for expanding document vectors, one may wish to expand vectors by means of a pre-assigned concept hierarchy. Various types of relationships between concepts may be specified and used in expansion procedures. The hierarchy consists of structured trees of concepts, in which each concept has a clearly defined "parent", plus a set of "brothers", and "sons". To designate looser connections, each term may be assigned a set of "cross-references". Expansion involves the introduction into the concept vector of the sons and/or brothers and/or father and/or cross-references of each term in the original vector. The selection of the method of expansion is by means of a weighting scheme, and is discussed in part 3.5.

To perform a hierarchical expansion the specification HIER a is given where a is either EXPAND or SHRINK. EXPAND indicates that the final vector is to contain the original vector plus the related concepts from the hierarchy; SHRINK indicates that the initial vector is deleted from the final vector, and only the expanded concepts are used. For example, if it is desired to generalize a vector, each concept could be replaced by its parent using HIER SHRINK; HIER EXPAND would merely augment each concept with its parent.

EXPAND a serves the same function as CONCON; a is either REQS, DOCS, or ALL and indicates respectively that requests, documents, or all input texts are to be expanded.

3.5. Vector Formation

The final vector is formed by combining all the concepts introduced from all sources with appropriate weights. The weighting parameters are numbers which are zero or greater (and may be greater than 1), indicating the weight associated with each unit occurrence of the given concept in the specified class. The weighting parameters are:

STEMWT x — x is the weight associated with word stem concepts. It should be a FORTRAN floating point number (e.g. 1.0, not 1);

STATWT x — x is the weight for statistical phrases. STATPR must have been specified when lookup was done; otherwise this command is ignored. If STATPR was specified, phrases may be ignored by giving STATWT 0.0;

SYNWT x — x is the weight associated with syntactic phrases;

COCOWT x — x is the weight associated with concept-concept expansion concepts.

The hierarchical weighting specifications are used to select the method of hierarchical expansion. Any number of expansions with arbitrary weights may be performed simultaneously. There are four weighting parameters, one for each possible mode of expansion. If the weight for a possible expansion mode is zero, that expansion is not performed. If the weight for a possible expansion is not zero, the expansion is performed and the concepts derived from it are given the indicated weight.

ROOTWT x — specifies the weight for expansion by parents;

BRANWT x — specifies the weight for expansion by brothers;

LEAFWT x — specifies the weight associated with expansion by sons;

CROSWT x — specifies the weight associated with expansion by cross-references.

Title and body differentiation is also performed by manipulating the weights. Usually, the weight associated with the body of the text is left at 1.0 and the weight of the title adjusted. However, to obtain only title processing, the user specifies BODYWT 0.0 and TITLWT 1.0.

BODYWT x — x is the weight associated with concepts from the body of the text.

TITLWT x — x is the weight associated with concepts from the title.

These parameters apply to all sources of concepts (phrases, etc.), including the expansion concepts (hierarchy and concept-concept).

LOGVEC — this specification causes all weights to be either zero or one. Weighting proceeds normally, except that after the computation of the weight, any nonzero weight is set to 1.

PRNVEC — causes the vectors to be printed for each document in the system. If concon or hierarchy expansions were specified, two printouts of vectors are made, one before and one after the expansion. In each printout the source of each concept and weight is shown in detail.

3.6. Request-Document Correlation

Once the final set of vectors is formed, the request vectors are correlated against the document vectors. The documents with high correlations are considered to be the computer-generated "answers" to the queries. These may now be printed. They can also be compared with a set of "relevant" documents, obtained from some other source. SMART can automatically compare the relevant documents with its answers, and evaluate the performance of the computer system against the outside standard.

The specifications which control this process are:

MODERD a — the correlation mode used for the request-document correlation is taken as the cosine mode or overlap mode, according as a is "COS" or "OVLAP" respectively. Cosine is the normal mode;

CUTRD x — the cutoff for request-document correlation is taken as x, a number between 0.0 and 1.0. Normal setting is 0.35;

ANSWER a — SMART can print out the answers to requests in three formats. If a is "SHORT" twelve-character identifiers are used for both requests and documents. If a is "MEDIUM", the full text of the request is used, and one line is used for the document identifier. If a is "LONG" the complete request and the complete document citations are printed for each answer. These formats assume, of course, that the necessary information is supplied with each document at read-in time (4.1);

SCORES — this specification causes the automatic evaluation procedure to be performed. A list of relevant documents must be provided, according to the format described in (4.4).

The output provided by the SCORES specification includes the following:

a) for each request, the fifteen documents with the highest correlations, and the values of those correlations;

b) for each request, the relevant documents, their ranks in the correlation list, and their correlations;

c) for each request, the normalized and un-normalized recall and precision measures;

d) the averages of the recall and precision measures over all of the requests used in this computer run;

e) a deck of the rank lists for each request, punched in the correct format for the MORVAL evaluation program (part 6.2).

3.7. Document-Document Expansion

After performing the initial request-document correlation, it is possible to augment the list of answers through a correlation of the documents with each other. Documents which exhibit a high correlation with documents already identified as "answers" can thus be retrieved. The specifications used in this process are:

DOCDOC — this specification indicates that a document-document correlation is to be performed;

MODEDD a — selects the correlation mode for document-document correlation as COS or OVLAP; COS is the normal mode;

CUTDD x — sets the cutoff for document-document correlation to x, a number between 0.0 and 1.0. The normal value is 0.5.

3.8. Other Specifications

DOCTAP — this specification indicates that looked-up documents appear on tape A6, in the format prepared by the MACTAP program (part 5.2). These documents will be read by the SMART programs, and are used as if they were included in the normal input tape;

AONE n — this specification sets the SMART tape AONE to FORTRAN logical tape number n. Normally n = 1;

ATWO, ..., BSIX — these specifications change the other tape specifications. Normal settings are shown in Table 1;

AGAIN — this specification indicates that another set of specifications follows this run on the input tape. SMART

proceeds to the next set after finishing the present run, instead of returning control to the monitor system;

PAGE n — this specification sets the initial page number to n. The normal setting is n = 1;

STOP or X — these synonymous specifications both indicate the end of the specification list. One of the two should be the last specification on the input cards; neither should occur anywhere else in the specification list.

A summary of the specifications is given in Table 2.

4. Data Input

After the specifications are submitted, the user must provide the data for the retrieval processing. This consists of requests, documents, and relevance judgments, in a typical run. They may be introduced in either English or in looked-up binary form, and documents may be introduced on A2, A6 or both.

Documents are divided into three categories, the requests, the texts (documents to be searched), and requests which are also texts. This last category consists of documents to be used as search requests, but which may also appear as answers to other search requests (normally, requests may not appear as answers to other requests). The last named document type might occur if a user has identified one document in his field of interest and wishes to see more documents on the same subject. All of these types may occur in natural language; in binary form, or both.

4.1. Natural Language Documents

Natural language input must be submitted before binary-form documents.

SMART Name	Normal Logical Unit	Harvard FMS Physical Tape Unit	Use
AONE	1	A1	Fortran monitor system tape
ATWO	5	A2	Monitor input tape - card input
ATHREE	6	A3	Monitor print tape - printed output
AFOUR	4	A4	Scratch tape - document identifications
AFIVE	9	A5	Scratch tape - document concept vectors
ASIX	11	A6	Document input
ASEVEN	12	A7	Scratch tape - used for sorting
AEIGHT	13	A8	Scratch tape - used for sorting
BONE	8	B1	Scratch tape - vectors for expansion
BTWO	2	B2	Scratch tape - correlations
BTHREE	3	B3	Program chain tape
BFOUR	7	B4	Monitor punch tape - card output
BFIVE	10	B5	Library - thesaurus, phrases,hierarchy
BSIX	16	B6	Scratch tape - used for sorting

Smart Tape Assignments

Table 1

*The only incompatibility with normal FMS operation is that B3 is not available as a scratch tape. This prevents compilations or assemblies when the program tape is used.

Specification	Value	Meaning	Normal Value
ENGTXT		Print texts of English input	off
NOTFND		Print words not found during the dictionary lookup	off
PUNCH		Punch out looked up documents in binary deck format	off
STATPR		Look up text for statistical phrases as well as word stems	off
SYNTAX		Search for criterion trees in text	off
NODECO		Print out details of criterion trees found	off
SYNANA		Print syntactic analysis of sentences analyzed	off
CONCON	n	Perform concept-concept correlations, iterating n times	off
MODECC	COS OVLAP	Choose mode of concept-concept correlation, cosine or overlap, (first iteration)	COS
MODE2C	COS OVLAP	Choose mode of later concept-concept correlations	COS
CUTCC	x	Set cutoff for first concept-concept correlations to x	0.60
CUT2C	x	Set cutoff for further concept-concept correlations to x	0.50
CONMIN	n	Do not include in the concept-concept correlations any concept numbers below n	0
CONMAX	n	Do not correlate in concept-concept correlations concept numbers above n	32000

Summary of Specifications

Table 2

Specification	Value	Meaning	Normal Value
EXPAND	REQS DOCS ALL	In concept-concept or hierarchical expansions, expand the requests only, or the documents only, or expand everything	ALL
HIER	EXPAND SHRINK	Perform a hierarchical alteration of concept vectors, either expanding them, or "shrinking" them (replacing the original concepts instead of augmenting them)	off
PCOCOR		Print the concept-concept correlations	off
LOGVEC		Do not compute any weights; non-zero weights are taken as 1	off
STEMWT	x	x is the weight of the concepts derived from word stems	1.0
STATWT	x	x is the weight attached to concepts derived from statistical phrases (note: all weights are relative; Usually STEMWT is left at 1.0, and other weights varied)	0.0
SYNWT	x	x is the weight attached to concepts derived from syntactic phrases	0.0
COCOWT	x	x is the weight attached to concepts derived from concept-concept expansions	0.0
ROOTWT	x	x is the weight attached to concepts derived from hierarchical expansion by parents	0.0
BRANWT	x	x is the weight attached to concepts derived from hierarchical expansion by brothers	0.0

Table 2 (continued)

Specification	Value	Meaning	Normal Value
LEAFWT	x	x is the weight attached to concepts derived from hierarchical expansion by sons	0.0
CROSWT	x	x is the weight attached to concepts derived from hierarchical expansion by cross-references	0.0
BODYWT	x	x is the weight attached to concepts derived from the body of the text (usually left at 1.0, unless only titles are to be processed, in which case set to 0.0)	1.0
TITLWT	x	x is the weight attached to concepts derived from the title of the text	1.0
PRNVEC		Concept vectors are printed for all requests and documents	off
MODERD	COS OVLAP	Choose mode of request-document correlation	COS
CUTRD	x	The cutoff for request-document correlation is set to x	0.35
ANSWER	SHORT MEDIUM LONG	The answers to the requests are printed in a condensed format, a moderately detailed format, or an expanded format, respectively	off
DOCDOC		The list of answers is expanded through document-document correlation	off
MODEDD	COS OVLAP	The mode of document-document correlation is set to cosine or overlap respectively	COS

Table 2 (continued)

Specification	Value	Meaning	Normal Value
CUTDD	x	The cutoff for document-document correlation is set to x	0.50
AONE	n	SMART tape AONE is set to logical tape n	1
ATWO ... BSIX	n n	SMART tapes ATWO, ... AEIGHT, BONE, ..., BSIX are set to logical units as specified (see Table 1)	
DOCTAP		Documents are read from ASIX in binary form, as well as from A2.	off
SCORES		An automatic evaluation is performed	off
AGAIN		Another set of specifications is read and processed after this run	off
STOP X		End of specification list; same as STOP	

Table 2 (continued)

Within the natural language documents pure requests must be submitted first, requests which are also texts next, and texts last.

Each document is preceded by an identifying card containing a * in column 1, a four character control word in cols 2-5, a blank in column 6, and a twelve character identifier in cols 7-18. Comments and identification are punched in 19-72, as desired. The control word is FIND for a pure request, LIKE for a request that is also a text, and TEXT for a pure text. The document follows the identifying card (the card with the asterisk in column 1) and continues on as many cards as are needed.

Basically, input text to SMART resembles typescript. For example, text may be punched anywhere in columns 1-72 of any number of cards. Any number of consecutive blanks are equivalent to one blank. A blank is assumed between column 72 of one card and column 1 of the next card.

The major differences from typescript are as follows:

a) An asterisk (*) in column 1, or two consecutive dollar signs anywhere ($$) are end-of-text signals. These indicators should not be used unless an end-of-text occurs. Normally, the end of a text is indicated only by the * in column 1 of the control card beginning the next text.

b) Periods not preceded by blanks are taken to be parts of abbreviations. Thus, a period meant to indicate the end of a sentence should be preceded by a blank.

c) SMART provides for the inclusion of up to 355 characters of identification for each text. This is used in the ANSWER LONG output option (see part 3.6). This identification should be punched at the beginning of the text on cards with a single $ in column 1. If a text begins with cards that contain a $

in column 1, SMART and all auxiliary programs ignore their contents, except that the first five (or fewer) such cards are collected as BCD identification. Usually, this consists of the title, author, and publication source of the document. The text proper is assumed to begin on the first card without a $ in column 1, and any further cards with $ in column 1 are treated as normal text. The identification may be omitted; in this case, the first card of the text must not have a $ in column 1.

d) The convention used to hyphenate a word between two cards is to place a minus sign (11-punch) followed by a blank at the end of the first part of the word. SMART ignores the remainder of the card, throws away the minus sign, and continues from column 1 of the next card. For example, hyphen- ate is properly hyphenated. Note that this rule makes the following construction illegal: "sub- and super-scripts." Also, note that normally hyphenated words which are broken between two lines at the hyphen must have a double minus sign to be properly recognized by the input programs; thus, Runge-- Kutta. Hyphenated words in the middle of a line are typed normally; thus, Runge-Kutta.

e) Hardware restrictions require that only upper-case letters be used. Special characters are treated as follows:

?	.QUE (preceded by a space),
!	.EP (preceded by a space),
"	/ (for both open and closed quotes. There should be no space before a close quote or after an open quote; similarly, parentheses and commas are spaced normally),
-	-DASH (11-punch minus sign followed by the word DASH, if a dash is meant; for hyphens see d) above),
'	'(8-4 punch, as in IBM Scientific character set H),
;	,. (not preceded by space),
:	.. (preceded by space, and ends a sentence).

Any other conventions could also be used, if the dictionaries were

constructed accordingly; but the foregoing are simple equivalents for the special characters.

The natural language documents must be on SMART tape ATWO. Bulk lookups from tape can be performed, however, by giving the specification ATWO 11 (defining ATWO as physical A6). In this case no binary documents can be submitted. Alternatively, ATWO can be defined as 12 (A7) or 13 (A8) in which case binary documents may still be submitted on A6 with a DOCTAP specification. Tape ATWO should not be moved to A4, A5, B1, B2, or B6; these tapes are used in the lookup.

4.2. Binary Documents

The binary documents are submitted on A2 and/or A6 following the English documents. If there are any binary documents on A2, they must be preceded with a card containing *ONLY in cols 1-5 and a blank in column 6.

As in the case of English documents, the pure requests are submitted first, followed by requests which are also texts, and finally concluding with the pure texts. The documents are preceded by *FIND, *LIKE, and *TEXT cards exactly as in part 4.1. The *ONLY card suffices to distinguish them from English documents. The binary cards which make up each document contain the identification punched at the beginning of the text, and the concept vectors obtained at lookup time. These documents are punched by the PUNCH specification, after the lookup. Thus, when binary documents only are being submitted, the library tape need not be mounted unless hierarchical expansion is required. The binary documents may be used with any title or phrase weighting (if STATPR was in effect at lookup time), and with any expansion options. If it is desired to change the

thesaurus or phrase list, documents must be looked up again. The punched-out decks include the appropriate *FIND, *LIKE or *TEXT card.

4.3. DOCTAPS

If the DOCTAP specification is given a tape of binary documents should be mounted on A6. This tape contains documents looked up in an earlier run, as is true also of the documents which follow the *ONLY card on A2. The tape on A6 is in a different, more compact, format, however, which is written by the program MACTAP (6.2) from the binary cards. The tape may contain several sets of looked-up documents. To select the set to be used, a *FILE card should be provided. The *FILE card contains *FILE in columns 1-5, a blank in column 6, and either an integer or an alphabetical file name in columns 7-18. If a file name is provided, the set of binary documents identified with that name on the *NAME card produced when the tape was written (part 6.2) is read from A6. If the integer n (punched anywhere in the field) is provided, the nth file is read. If no *FILE card is provided, the physically first file on A6 is used. The *FILE card is placed on A2 following the specifications, but preceding the *STOP card and the *ONLY card (if there is one). The *FILE card must not appear in the middle of an English document, however, nor between a *TEXT, *FIND, or *LIKE card and the following document.

4.4. Relevance Judgment Data

After all documents are read, the next input card must have *STOP in cols 1-5 and a blank in column 6. This card must occur exactly once on A2 for each set of instructions, following all English and binary documents (if any). It must also occur at the end of each set of documents on A6.

Following the *STOP card, the relevant document data needed for the automatic evaluation system must be submitted, if SCORES is specified. If SCORES is not specified, relevant documents are omitted. The cards containing the relevance data are preceded and followed by a card with *RELS in columns 1-5 and a blank in column 6. For each request, a card is punched with the request identifier in cols 1-12, and the number of relevant documents in cols 13-16, right adjusted. This card is followed by a set of cards specifying the relevant documents. Each relevant document identifier is punched in cols 1-12 of a separate card. Columns 13-15 should be left blank, as these are reserved for a future extension of the evaluation program to include degrees of relevance.

Relevance judgment data may be placed on either A2 or A6. If relevance judgments are given, they must occur as one continuous deck of cards, surrounded by *RELS cards, placed after the *STOP card. Exactly one set of relevance judgments should be supplied for each request (i.e., documents preceded by *FIND or *LIKE cards). These relevance judgments may be on either tape, in any order. All document and request identifiers, however, must be spelled in exactly the same way on the relevance judgment cards as on the *FIND, *LIKE, and *TEXT cards. Relevance judgments may be placed on A6, whether or not SCORES is requested when the tape is read, but if SCORES is not specified and AGAIN is specified, there should be no relevance judgments on A2.

4.5. Other Instruction Cards

To place comments on the print tape, the *NOTE card is used. This card contains a *NOTE in columns 1-5, blank in column 6, and comments to be printed and ignored in columns 6-72. Printing is done off-line.

To determine the time taken by a program run, a card with *TIME in columns 1-5 and a blank in column 6 is placed on A2. When this card is read, the elapsed time is printed. The printer clock (IBM RPQ 78054) is required if the timing printout is to operate.

A list of all the SMART instruction cards with a * in column 1 is given in Table 3.

5. Tape Preparation Programs

SMART retrieval runs make use of a set of pre-written tapes, to avoid inconveniently large input decks. There is no loss of flexibility or capacity in the use of tapes. Up to three special tapes are used: the library tape, the document tape, and the program tape.

The program tape contains the binary programs used by the SMART system. It is mounted on B3. The document tape, on A6, contains looked-up document collections to be processed. The library tape, which goes on B5, contains the thesaurus, phrase dictionaries, grammar, and hierarchy.

5.1. Writing a New Library Tape

The library tape contains six files. These are the thesaurus, the statistical phrase list, the syntactic suffix list, the English grammar, the criterion trees (syntactic phrase list), and the hierarchy. A set of programs in two chain links (numbered 101 and 102) is used to write a library tape. The programs in link 101 update the thesaurus file; the programs in link 102 write the remainder of the tape. Link 101 automatically calls link 102. The programmer may omit link 102, however, if nothing but a thesaurus is desired on the library tape (this is usually the case with null dictionaries).

Cols 1-6	Cols 7-18	Meaning
*FIND	Request name	Precedes a pure request for the system
*LIKE	Document name	Precedes a text which is also a request; i.e., a document which is both searched for and searched on
*TEXT	Document name	Precedes a pure text, i.e. a document which is only searched for
*STOP	Anything	End of documents submitted on this tape
*RELS	Anything	Precedes and follows deck of relevance judgments
*NOTE	Anything	Comments
*TIME	Anything	Finds out the time
*ONLY	Anything	Indicates end of English language documents and beginning of binary documents
*FILE	Collection name or integer	Identifies which collection of documents on A6 is to be read in to the system

SMART Instruction Cards

Table 3

The new library tape is written on B5. If an old tape is being updated, it is placed on A6. New data is submitted on cards, on A2, except for a new grammar, which is presented as a tape on A5.

5.1.1. Thesaurus and Suffix List Formation

The thesaurus is the first file on B5 to be written. It contains a suffix list used in the lookup, followed by the thesaurus records themselves. The thesaurus and the suffix list may be introduced from cards (on A2), or from an older library tape on A6.

The first data card for the update programs specifies whether the thesaurus and suffix list are on A2 or A6. This card contains a word, left-adjusted in columns 1-6, and followed by trailing blanks until column 7, which identifies those data which are on tape A2. If this word is BOTH, the thesaurus and suffix list are assumed to be on A2, and tape A6 is not read. If it is desired to skip over (without reading) a thesaurus file on the old library tape, columns 7-12 of the BOTH card should contain the word SPACE followed by a blank. If the suffix list is being submitted on cards, while the thesaurus is to be copied from A6, columns 1-6 of the first card should contain the word SUFFIX. The word THES identifies the thesaurus as being on cards, while the suffix list is taken from the old tape. If both the thesaurus and the suffix list are on the old library tape, COPY is given as the control word.

If the first data card contains either SUFFIX or BOTH, the next data deck must be the suffix list. Each suffix is punched on a separate card, beginning in column 1, and followed by blanks. Each suffix should be assigned a distinct sequence number, less than 256, which is punched right-adjusted in columns 13-15. The suffix list ends with a card con-

taining ZZZZZZ in columns 1-6. The suffix list may appear in any order, provided that the first suffix begins with "e".

If the thesaurus is introduced from cards (control word BOTH or THES), it follows the suffix list (if any) immediately. If there is no suffix list, it follows the first data card. The thesaurus may contain any number of words. Each word is punched on a separate card. Since a suffixing routine is available, only the word stems should be entered in the dictionary. The suffixing routine will accept multiple suffixes, and also recognizes that words may double their final consonant, change a final "y" to an "i", or drop a final "e" before adding a suffix [7]. This permits a highly accurate lookup in which almost all derivatives of most English words can be identified automatically [12]. Of course, the extremely variant forms (such as "mice", "geese", etc.) must be entered in the dictionary as separate words. It is sometimes necessary to enter complete forms of words to preserve important distinctions. For example, "programmer" is readily identified from "program" + "m" + "er", but if it is desired to distinguish a programmer from a program, both forms should be entered. The dictionary lookup procedure may always be used as a full paradigm dictionary by entering every form of a word in the thesaurus, at the expense of a great deal of keypunching. Words entered explicitly take precedence in the lookup over possible stem-and-suffix combinations.

Each stem to be entered in the dictionary is punched left-adjusted in columns 1-24 of a card, with trailing blanks.

With each word are associated between one and six concept numbers. These are punched right-adjusted in five column fields, starting in column 25 and extending to 54. The first concept number is punched in

25-29, the next in 30-34, then in 35-39, and so on. If there are less than six concept numbers, the remaining fields should be left blank. No concept number should exceed 32767. Concept numbers above 32000 (and the concept number 0) are reserved for non-significant words. The concept numbers 1-32000 are significant concepts, used in correlation process.

Up to eight syntactic codes may also be punched for each stem. They are punched in columns 55-78 in three-column numeric fields, right adjusted within each field but using the leftmost columns first. Each syntactic code is a number, less than 256. As currently implemented, these numbers correspond to partial stem homographs for the Kuno syntactic analyzer. The correspondence between syntactic code numbers and partial homographs is shown in Table 4. The homograph is completed by combining the partial stem homograph with the partial suffix homograph of the syntactic suffix list (5.3.1) as explained in [13].

The word stems on the input cards should be in correct BCD alphabetical order. They must be correctly ordered by the first letter, as the programs are currently arranged: the degree of order necessary depends on the size of the dictionary. The thesaurus must end with a card containing a 0-5-8 multipunch in column 1, and *END* in cols 2-6.

5.1.2. Statistical Phrase Dictionary

The second file on the library tape contains the first of the phrase dictionaries, the statistical phrase dictionary. The first card of the statistical phrase deck (following the last card of the thesaurus update section) is a control card specifying the use to be made of the old file on tape A6. Columns 1-6 of this card should be either REPLAC, indicating

1	ADJ	21	BEOS	41	OI20	61	HVI	81	QUE		
2	ADK	22	BEOY	42	OI30	62	HVP	82	RL1		
3	ADL	23	BGOS	43	OT10	63	IAD	83	RL2		
4	ADM	24	BIO	44	OT20	64	IAV	84	RL3		
5	ADN	25	BPO	45	OT30	65	IPN	85	RL4		
6	ADØ	26	BRO	46	OT40	66	IPØ	86	RL5		
7	ADP	27	CCØ	47	OT50	67	NAD	87	RL6		
8	ART	28	CIF	48	OT60	68	NØ4C	88	TITS		
9	AUXC	29	CMA	49	OT601	69	NØ4S	89	TØIS		
10	AUXP	30	CØ1	50	OT70	70	NØUO	90	XCØ		
11	AUXS	31	CØ2	51	OT701	71	NØVO	91	YCØ		
12	AV1	32	CØ3	52	HAVC	72	NUMO	92	NØUS		
13	AV2	33	CØ4	53	HAVP	73	PRD	93	VT1P		
14	AV3	34	CØ5	54	HAVS	74	PRE	94	VI1P		
15	AV4	35	CØ6	55	HP1	75	PRNC	95	NØUP		
16	AV5	36	CØ7	56	HP3	76	PRNP	96	NØUC		
17	AV6	37	CØ8	57	HP4	77	PRNS	97			
18	AV7	38	CPR	58	HP5	78	PRØ	98			
19	AV8	39	DØI	59	HPP	79	PRZC	99			
20	BEOP	40	OI10	60	HVGS	80	PRZP				

100	VI1P	110	VI2P	120	VI3P	130	VT1P	140	VT2P	150	VT3P
101	II1	111	II2	121	II3	131	IT1	141	IT2	151	IT3
102	VI1S	112	VI2S	122	VI3S	132	VT1S	142	VT2S	152	VT3S
103	VI1C	113	VI2C	123	VI3C	133	VT1C	143	VT2C	153	VT3C
104	PI1	114	PI2	124	PI3	134	PT1	144	PT2	154	PT3
105	GI1S	115	GI1S	125	GI3S	135	GT1S	145	GT2S	155	GT3S
106	RI1	116	RI1	126	RI1	136	RT1	146	RT2	156	RT3
107		117		127		137		147		157	
108		118		128		138		148		158	
109		119		129		139		149		159	

160	VT4P	170	VT5P	180	VT6P	190	VT6P1	200	VT7P	210	VT7P1
161	IT4	171	IT5	181	IT6	191	IT6 1	201	IT7	211	IT7 1
162	VT4S	172	VT5S	182	VT6S	192	VT6S1	202	VT7S	212	VT7S1
163	VT4C	173	VT5C	183	VT6C	193	VT6C1	203	VT7C	213	VT7C1
164	PT4	174	PT5	184	PT6	194	PT6 1	204	PT7	214	PT7 1
165	GT4S	175	GT5S	185	GT6S	195	GT6S1	205	GT7S	215	GT7S1
166	RT4	176	RT5	186	RT6	196	RT6 1	206	RT7	216	RT7 1
167		177		187		197		207			
168		178		188		198		208			
169		179		189		199		209			

Partial Stem Homographs
(See Mathematical Linguistics and Automatic Translation, Report No. NSF-9, Vol. I)

Table 4

Note: Slashed Ø stands for letter "Oh";
Normal 0 stands for numeral zero.

that new cards on A2 replace the old file on A6 which is skipped, or IGNORE, indicating that new cards are on A2 and tape A6 is to be ignored, or COPY, indicating that the old phrase dictionary on A6 is to be copied unchanged to B5.

If REPLAC or IGNORE are specified, the control card must be followed by the cards for the phrase dictionary. Each phrase is punched on a separate card. The concept number of the entire phrase is punched right-adjusted in columns 1-5. The component concept numbers are punched right-adjusted in six-column fields, using the leftmost fields first. Thus, the first component goes in columns 6-11, the next in 12-17, the third component (if any) in 18-23, and so on.

Statistical phrases must have two components, and may have up to six. Non-significant concept numbers may be used; for example, one could find the phrase "exclusive or" by searching for the components "exclusive" and "or" even if "or" had a non-significant number (as it usually does).

The statistical phrase dictionary may be in any order, but the last card must contain NOMOR in columns 1-5 and a blank in column 6.

5.1.3. Syntactic Suffix List

The second file on the library tape contains the partial homograph codes associated with the suffixes. One control card is required by the library update programs; it contains one of four codes in column 1-5. The possible codes and their meaning are:

(ORIG A deck of suffix cards follows in the format described below. Generate a file of these suffixes, ignoring tape A6.

(DUPL The file on A6 is copied unchanged to B5. No other cards

are submitted for this file.

(MERG Merge the new suffixes which follow with the old list.

(SKIP Same function as (ORIG, except that A6 is spaced over the old file.

The suffix cards required for the (ORIG, (DUPL, and (MERG options are punched as follows. Each card contains one suffix identified by its number, and the list of partial homographs for this suffix. The suffix number (associated with the suffix by the cards described in part 5.1.1) is punched anywhere in columns 1-6, and the various possible partial or complete homograph codes are punched in six-column alphabetical fields beginning in column 7, using the leftmost fields first.

The partial homographs are alphabetic, with the unneeded characters replaced by zeros. For example, the Kuno homograph for a plural noun is NØUP. A noun stem, in the SMART dictionary, is given the syntactic code number 70, corresponding to NØUO (note as before that O is "zero" while Ø is the fifteenth letter of the alphabet). The suffix "s" has a partial homograph OOOPO. When stem and suffix are combined, the complete homograph NØUP is formed. The stem RECTI, to take another example, has the syntax number 043, which corresponds to OTlO. When combined with the suffix FIED, which has the suffix homograph VOOC (among others), a correct homograph VTlC is obtained for RECTIFIED. The remaining homographs are obtained from the other suffix homographs; they are ADJ (obtained completely from the suffix) and PTl (OTlO + POO O = PTl).

The partial homographs of some sample suffixes are shown in Table 5.

5.1.4. The Condensed Grammar File

The condensed grammar file follows the syntactic suffix list on the

Suffix	Homographs
ED	VOOCO POO O ADJ NOVC
S	NOUP VOOSO
ES	NOUP VOOSO
LY	ADJ AVI
ING	ROO O GOOSO NOVS ADJ
Y	NOVS ADJ
ION	NOUS
IONS	NOUP
OUS	ADJ NOVC
ER	NOVS ADK AV6
EST	ADJ NOVC
ENT	ADJ NOVC
ERS	NOUP
ABILITIES	NOUP

Some Partial Suffix Homographs

Table 5

library tape. It is used by the Kuno multiple-path syntactic analyzer. This analyzer is discussed in [9]. The grammar tape itself is not actually written by any program in the SMART package; a tape containing the grammar is prepared by the analyzer programs operating independently. This tape is then copied onto the SMART library tape.

The card following the syntactic phrase list directs the tape copy. If it begins with COPYA5 in columns 1-6, a new grammar tape on A5 replaces the old grammar. If it begins with COPYA6, the old grammar is used. If UPDTA5 appears in cols 1-6, a new grammar is copied from A5 and the old grammar is skipped on A6.

The SMART programs have been run with a grammar modified to accept noun phrases and prepositional phrases. This permits the syntactic analysis of titles [14]. However, as noted above, the syntactic analysis routines are generally in disuse because of timing problems, since the new, faster analyzer is not yet available.

5.1.5. The Criterion Tree File

The fifth file of the library tape consists of the criterion trees, a phrase dictionary in which phrases are defined by a complete set of structural/semantic/syntactic specifications. The overall control cards recognized by the update editing criterion tree routine are:

/COPY in columns 1-5, with an optional integer beginning in column 7. The indicated number of trees are copied from A6 to B5. If no number is given, the whole criterion tree file is copied.

/SKIP in columns 1-5, with an optional integer beginning in column 7. The indicated number of trees on A6 are skipped (if no number is given, the whole file is skipped).

/EDIT in columns 1-5, with an optional integer beginning in column 7. The indicated number of trees are copied from A6 to B5 with the trees specified by the deletion requests (which follow this card) removed from the file. These deletion requests are cards with right parentheses in columns 1 and 2, and with a serial number in columns 7-12 and/or a BCD identifier in columns 13-18. Up to 30 of these delete requests may be given. Any criterion tree on the tape which matches the information on the delete card (either the identifier or the serial number, if only one is specified, or both the identifier and the serial number if both are given on the delete card) is removed from the file.

/WEOF in columns 1-5. This terminates the processing of the criterion trees. The file on B5 is terminated and A6 is spaced over any remaining trees. This should be the last control card affecting the criterion trees.

/ADDL This specification causes trees to be read from A2, encoded as required by the search programs, and written on B5. The format required for the specification of criterion trees is given below.

5.1.5.1. Criterion Tree Input Format

Each definition begins on a new card. Basically, six pieces of information are required to define a criterion tree. These are:

a) the tree name and serial number,

b) the output concept numbers to be assigned to the sentence if the tree is found,

c) the place in the sentence where these concepts are attached,

d) the dependency relations between nodes (the structure of the tree),

e) the syntactic values associated with each node that must be matched,

f) the semantic values associated with each node, which must be matched.

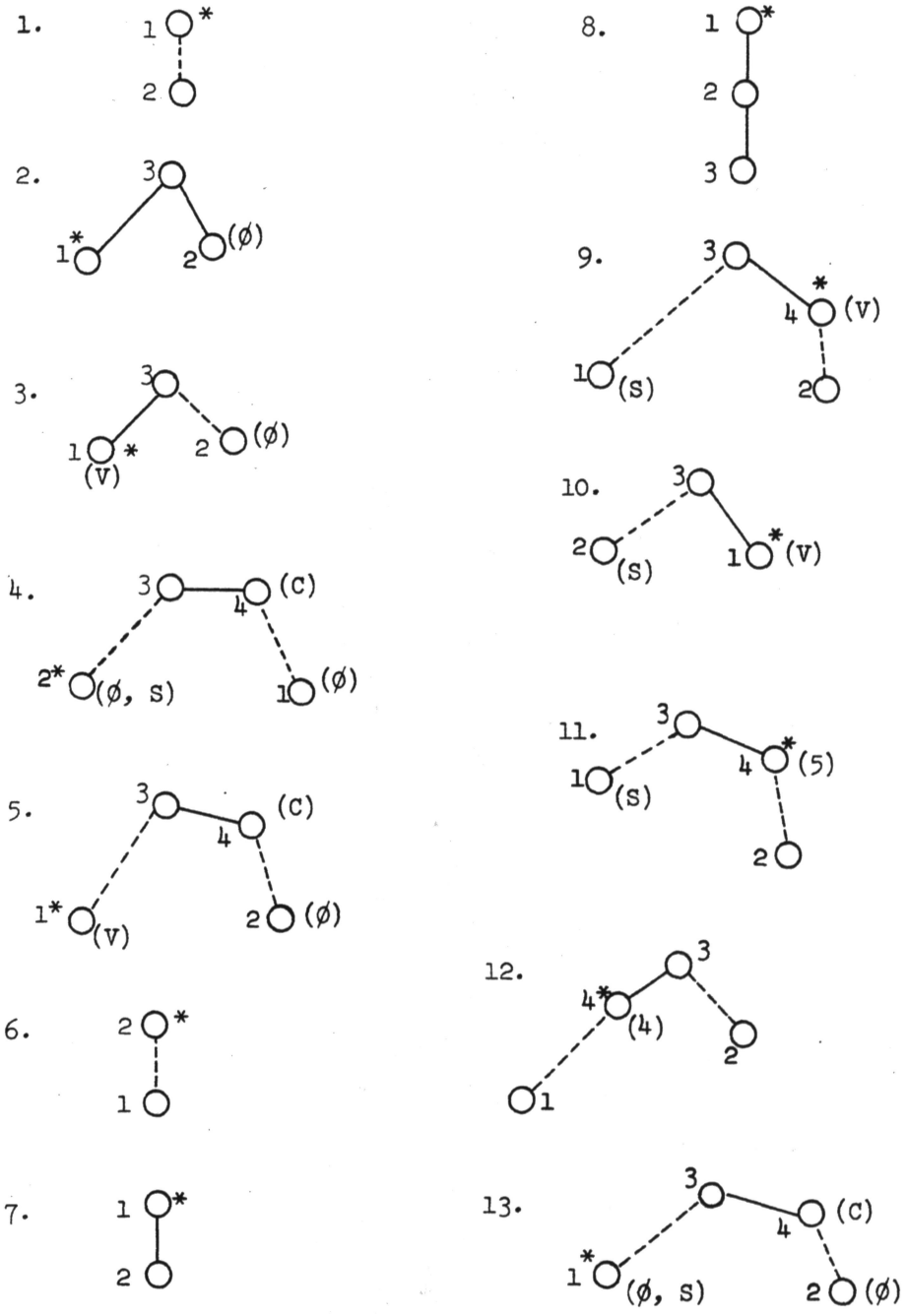
Normally, the data under c), d) and e) are supplied by reference to a standard "tree type". A table of tree "types" exists in the program, represented in Table 6. New types may be defined, however. Trees may also be defined completely on the input cards. In fact, one may simultaneously define both trees and types with an input card. Usually, however, only the output concept number, the semantic data, the tree type, and the name and serial number are assigned by the user. The format of the criterion tree input cards, going from left to right, is as follows:

a) columns 1-6 contain six BCD characters. If this is the first card of a tree, this field contains the name of the tree. This is the BCD identifier used in the /EDIT control card. If this field contains five blanks followed by an asterisk, this card is a continuation card from the preceding card. Column 1 should not contain a /, except for the last input card of the criterion tree deck, which must contain ////// in cols 1-6.

   If a type, rather than a tree, is being defined, the BCD name field is ignored. However, it must be non-blank, non-zero, and satisfy all the other requirements as usual.

b) the output concept number field(s) follow. This may be omitted if a type only, and no trees, are being defined. The contents of this field are the concept numbers to be used as output, and the positions in the sentence (the nodes) where they are to be inserted. More than one output concept number field is allowed; each field, including the first output concept number field, must be preceded by an equals sign (=). Each field is defined as follows:

   i) if the output concept number is a specific integer, it must be



Standard Tree Types

Table 6

Solid lines indicate direct dependence; dotted lines indicate indirect dependence. The asterisk denotes "key" nodes; i.e. the nodes to which the output concept number is attached when a match is found.

first in the field, and only one such integer is allowed. For example, to specify 285 as the output concept number, one writes = 285 for the output concept number field;

ii) if no definite integer is given, at least one subfield consisting of the letter "N" followed by an integer must appear. Each such integer identifies a node in the tree whose concept number is used as an output concept number. No nodes numbered above 15 may be referred to in this way;

iii) normally, a pre-assigned "key" node in the tree receives the output concept number. In Table 6 this key node is marked with a * for each type. If the user wishes to assign the output concept number to another node or nodes he may do so by punching a period followed by the number of the node to which the concept number is assigned. Any number of periods followed by integers (the integers must be smaller than 15) may be punched. A period not followed by an integer suppresses the weight assignment to the key node. For example, if it is desired to change the assignment from the key node to node number 3, the user would punch .3. in this subfield;

iv) normally, the output concept numbers are given equal weights, assigned in such a way that the total weights of each phrase are equal. If SYNWT 1.0 is specified, the total weight of each phrase will be equal to the weight of one word occurrence (assuming the normal STEMWT 1.0). However, the user may punch a subfield consisting of an asterisk followed by an integer. The output concept number of this field will then be assigned a weight specified by this integer. The integer may be zero; it must be less than seven. Normally, weighting with one output concept number per phrase corresponds to an integer specification of 4. To halve the weights, specify 3; to double them, 6; and to triple them, 7. A zero specification eliminates this phrase.

Thus, to define an output concept number of 285, and to attach it to node number 3 instead of the keynode, and also to double its weight, the user punches =285.3.*6 as an output concept number field. If, in addition, the concept number on node 2 is to be moved onto nodes 4 and 5 with half weight, another output concept number field would be added, the total reading:

=285.3.*6=N2.4.5.*3

c) it is also necessary to supply the conditions specified for each node. For each node exactly one node condition field exists, except that conditions need not be specified for the last node (and adjacent nodes) if a definition using a tree type is employed, and the syntactic and structural criteria supplied by the type are all that is needed for these trailing nodes.

The first node condition field begins immediately after the output concept number fields. Each node condition field is followed by a slash, (/) except for the last node condition field which is followed by a dollar sign ($). If the tree is being defined according to a standard type, only the semantic information need be supplied in the node condition fields. If a type is being defined, however, or if a tree is being defined without the use of a type, all information must be supplied for each node. In this case, there must be exactly as many node condition fields as there are nodes in the final tree.

If structural data is being specified, all but exactly one node condition field should begin with either an I or a D followed by an integer. The node whose condition field does not begin with an I or D is taken as the root of the criterion tree. A node whose condition field begins with "In" must match a node which is descended from (direct or indirect ancestor) the correspondent of node number n. A node whose condition field begins with "Dn" must match a node which is the direct (first-generation) descendant of the node which matches node n. "Descend" is understood in the

sense of "being syntactically dependent upon, according to the syntactic analysis used for this sentence."

The semantic (and syntactic, if needed) data now follow; or, if this tree is being defined using a standard tree type, these data begin the field. The node condition field, except for the In or Dn subfield, is composed of a set of "relations", each "relation" being a set of one or more "relation generators". Each relation generator is either a concept number or a syntactic role code. A concept number, indicating a semantic condition, is punched as an explicit integer. A syntactic role code, indicating a syntactic condition, is punched as a letter enclosed by apostrophes. Thus, the relation generator for concept number 285 is simply 285; the relation generator for a verb is 'V'. The syntactic role codes and their meanings are shown in Table 7. If a tree node is to match a sentence node, at least one relation generator in <u>each</u> relation must correspond to a property of the word. Relation generators are punched separately by commas, and all the relation generators in each relation are enclosed with parentheses. For example, the node condition field I3(8,16)('Ø','S') can only match a word which:

i) depends directly or indirectly on the word matching node number 3;

ii) corresponds to either concept number 8 or concept number 16 in the thesaurus;

iii) functions either as an object or a subject in the sentence.

On the other hand, the condition field (8)(16) would only match a word which would correspond to both concept 8 <u>and</u> concept 16 in the thesaurus.

The node condition fields correspond to nodes in sequence counting from left to right. Thus, the node condition fields (26,78)/(95)/(22,105) indicate that node 1 must match a word in concept 26 or 78; node 2 must match a word in concept 95; and node

Letter	Meaning
1	Declarative sentence
2	Interrogative sentence
3	Imperative sentence
4	Subject clause
5	Object clause
6	Complement clause
7	Adjective clause
8	Adverbial clause
A	Adjective
C	Complement
D	Adverb
E	Adverbial noun phrase
G	Gerund
M	Participle
ø	Object
P	Phrase
R	Phrase or clause introducer (preposition or conjunction)
S	Subject
V	Verb
X	Auxiliary Verb
+	Conjunction (and/or/but)
,	Comma
.	Period
=	Question mark

Syntactic Role Codes

Table 7

Footnote to Table: see section I of Reference 8, page 127 (Table 25)

3 must match a word with either concept 22 or 105. The structural and syntactic data are supplied, as usual, from the tree type to be specified later.

d) following the dollar sign ($) ending the node conditions fields, a final, miscellaneous field appears. This field specifies whether trees or types are being defined, what types if any are being used in the tree definitions, and what the serial number(s) of the output tree(s) are.

If trees, rather than types, are being defined, the dollar sign ($) is followed with a set of tree specifications, separated by commas. Each specification consists of a type number, identifying the standard type which supplies the syntactic and structural data, and a serial number code. The type number is simply an integer, identifying the tree in Table 6. It is followed immediately by the serial number code, which is one of the following:

i) + (plus sign) indicating a serial number one greater than the last serial number assigned;

ii) * (asterisk) indicating a serial number equal to the last serial number assigned;

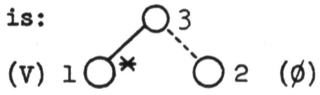
iii) / (slash) followed by an integer, which assigns this integer as a serial number.

If the syntactic and structural data are supplied with the tree definition, instead of from a type, the type number in the tree specification should be explicitly set to zero. Any number of tree specifications may be given, separated by commas, to define several trees at once. For example, one can simultaneously define trees to represent "information retrieval is ..." and "retrieving information by ..." in this manner.

If a new type is defined, this last field should end with an integer followed by either "T" or "P". This declares the structure and syntactic conditions defined on this code to be a standard tree type, numbered with the integer provided. This rule applies

even if the integer was previously in use to identify another tree; the old tree is lost in this case. If the letter "P" is used, in addition to specifying a definition valid for this run, a deck of binary cards is also punched which may later be added to the deck of the subprogram TRECND in link 102 to permanently alter the standard tree types. The "T" or "P" should be followed by a set of integers, separated by commas, which identify the "key" nodes of the new tree type. Usually there is only one key node.

e) the following additional format requirements should be noted. Blanks are completely forbidden except in the BCD name field. The occurrence of a blank ends the definition. To continue a definition from one card to the next, a minus sign (-) should be punched as the last non-blank character of the first card, and the next card should be started with five blanks and an asterisk in columns 1-6. Punching of the tree definition should be resumed in column 7.

Consider the following examples. Assume that "information" is represented by concept 114, "data" by 53, "retrieval" 26, and "information retrieval" is to be defined concept 301. Note that tree type 1 is :  and type 3 is: 

To define tree type 3, one could write:

```
TYPE3 D3('V')/I3('Ø')//$3T1
```

and to define type 1:

```
TYPE1 /I1$1T1
```

This tree would match all the following sentences if the appropriate concept numbers are assigned to the nodes (numbering the trees 10 and 11, and naming them INFRET):

```
INFRET=301(26)/(53,114)$1/10,3+
```

Information retrieval is useful,

Data retrieval is useful,

They retrieved data by computer techniques,

They retrieved information by computer techniques.

To match "They retrieved information by computer" but assign the output concept 301 to "information" instead of to "retrieved" one writes

INFRET=301.2.(26)/(53,114)$3/10

To define tree type 3 and the original tree simultaneously, one writes

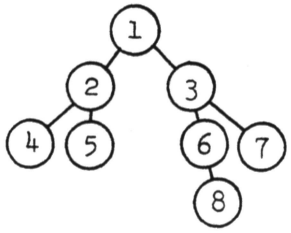
INFRET=301D3('V')(26)/I3('0')(53,114)/$0/10,3T1

5.1.6. Hierarchy

The hierarchy is punched in a comparatively simple format. Each concept number is punched on a separate card, and represents a single node in the hierarchy. Indenting is used to indicate descent in the hierarchy. More exactly, the "level" of a node is defined as the number of the column on the IBM card in which the units digit of the concept number is punched. The parent of each node is defined as the last preceding node on a lower level, i.e. with its units digit punched further to the left. If a node is furnished with cross-references to other nodes, these are punched on the same card, to the right of the node number. There may be any number of cross-references, punched anywhere on the card (separated by blanks). All punching is restricted to columns 1-70. If additional space is needed, any character may be punched in column 72. This causes the next card to be interpreted as a continuation card.

It is esthetically pleasing, although not necessary, to punch all brothers on the same level (in the same columns). For example, the following structure (with node number 2 cross referenced to node 7 and vice

versa):



could be punched either as:

```
1
  2   7
    4
    5
  3
    6
      8
    7 2
```

or as:

```
1
     2 7
           4
        5
   3
                 6
                   8
     7  2
```

If the hierarchy consists of several bushes, rather than one connected tree, bushes are separated by single blank cards. Two consecutive blank cards end the hierarchy.

5.2. The Document Tape

To avoid the necessity of submitting document collections on cards for each retrieval run, a prewritten tape of binary documents can be used. These tapes are written by a program called MACTAP. The control cards for MACTAP contain asterisks (*) in column 1, and control words left-adjusted with trailing blanks in columns 2-6. Data cards are submitted on A2 and the tape is written on A6.

The first data card may be the control card *START. This card is used if additional data are being added to a previously written document tape. If it is omitted, it is assumed that a blank tape is mounted on A6.

Any number of sets of documents may be submitted in one run. Each set of documents is preceded by a *NAME card, which contains a twelve character identifier in columns 7-18. This identifier represents the collection name which is used in the search initiated by the *FILE card (see part 4.3). The documents are submitted on cards, in looked-up form. The order is the usual: first any pure requests (identified by *FIND cards), then any combination request/texts (identified by *LIKE cards), and finally texts, identified by *TEXT cards. A *STOP card follows each set of documents. A deck of relevance judgments, as in part 4.4, may be placed between *RELS cards after the *STOP card.

After each set of documents and relevance judgments, another *NAME card and another document collection may be added. The end of the run is marked by a *FINISH card.

5.3. The Program Tape

The SMART program tape is a normal FMS chain tape. To write it, the links to be included are submitted on binary cards, with a blank tape on B3. The FMS loader will write a chain tape automatically, which may be saved and reused.

Currently, SMART has 14 chain links, as shown in Table 8. Tape-writing programs operate separately. All links need not be included on the program tape, but if any links are omitted some options will be unavailable. It is necessary to resubmit the programs on cards and have the loader write a new chain tape to make any changes in any of the programs.

6. Auxiliary Programs for Use with the SMART System

Several programs, although not directly a part of the SMART system, are

Link No.	Function
1	Lookup in thesaurus and statistical phrase dictionary. Printing of English texts
2	Printing of words not found in thesaurus
3	Formation of complete homographs (lookup in syntactic suffix list)
4	Syntactic analyzer, link 1; formation of binary sentence tape
5	Syntactic analyzer, link 2; analysis, formation of binary analysis tape
6	Syntactic analyzer, link 3; formation of edited analysis tape
7	Criterion tree matching routine; lookup of syntactic phrases
8	Vector formation, combining all sources of concept numbers, weighting of vectors; correlation of requests against documents
9	Concept-concept correlation; formation of concept-concept expansion matrix
10	Hierarchical and concept-concept expansions; formation of new document vectors
11	Document-document correlation, and expansion
12	Sorting of request-document correlations
13	Printing of answers to requests
14	Evaluation of run

SMART Chain Links

Table 8

frequently used for auxiliary functions. These are described in the present section.

6.1. THES

THES is a program to form null dictionaries from a collection of English text. It also prints out frequency counts and listings as a by-product. It includes a suffixing routine.

THES requires an A2 one control card. This card contains three integers. The first integer, punched right-adjusted in columns 1-5, specifies the maximum number of concepts (words) to be included in the null dictionary. The second integer, punched right-adjusted in columns 6-10, specifies the minimum number of occurrences in the collection that any word in the dictionary may be expected to exhibit. These two numbers permit the user to control the size of the null dictionary. For a complete null dictionary, the first number should be very large and the second number should be 1. The third number is punched right-adjusted in columns 11-13 and specifies the tape on which the document collection is located. If this field is blank, tape 5 (the input tape) is assumed.

The collection is placed on the specified tape in normal SMART format (4.1), with documents preceded by *TEXT cards only. *FIND cards, and *LIKE cards should not be used. Of course, since no searches are made during thesaurus construction, the requests may be labeled *TEXT, without problems, if it is desired to include them in the counts. A *STOP card ends the collection.

6.2. MORVAL

MORVAL is a program to compute additional evaluation data for a set of

requests and methods. MORVAL accepts as input a set of decks of rank positions punched by the SCORES specification at run time. The program produces for each method and each request the top fifteen documents, and the ranks of the relevant documents. All documents are abbreviated to three-character identifiers to permit a compact output format. Thus, to use MORVAL properly the twelve-character identifiers used for the documents should be chosen so that the first three of these characters are adequate to uniquely specify a document. MORVAL also produces normalized and unnormalized evaluation measures, and recall-precision graphs of two types (averaged over requests, and cumulated-type graphs). MORVAL is also capable of evaluating "merged methods", where the results of several methods are merged.

MORVAL is described in reference [15], where complete instructions for using the program are given.

6.3. SOCCER

SOCCER is a concordance generating program. All occurrences of each word in an input document collection can be listed with their context. Such a concordance greatly simplifies many tasks of word use analysis. SOCCER contains full facilities for suppressing concordances on unwanted words; or restricting the concordance to selected words. It also provides a variety of statistics about the collection of text processed.

Because of the simpler nature of the processing, SOCCER will function with virtually any input format for the text. In particular, the SMART formats are satisfactory, and are convenient for some of the format options of SOCCER.

SOCCER is a relatively fast program; a concordance prepared for a

collection of 110,000 words took about forty minutes. SOCCER is fully described and complete instructions given for use in [16]. It has been distributed by SHARE as distribution no. 3407.

7. A Sample Input Deck

Note: Normal typing represents comments; UPPER CASE UNDERLINED represents BCD input cards; lower case underlined represents binary input cards.

```
* JOB,630201,5MIN,5000,LESK
```

This is the Harvard FMS job card format.

```
* XEQ
```

Monitor control card.

```
* PLEASE MOUNT TAPES ...
*   SALTON 25 ON B3 RING OUT 800 BPI
*   SALTON 44 ON A6 RING OUT 800 BPI
* PAUSE (THANK YOU)
```

These cards instruct the operator to mount the program tape, Salton 25, and the document tape, Salton 44.

clchn - two card binary program

This program is equivalent to the compiled version of the Fortran statement CALL CHAIN (1,3). It brings in the SMART programs from the program tape.

* DATA

Monitor control card.

DOCTAP ANSWER SHORT SCORES MODERD COS CUTRD 0.35 PRNVEC

STATWT 2,0 X

These instructions cause the programs to

a) read input data from a document tape mounted on A6;

b) to print only twelve-character identifiers for the request and documents when answers are obtained;

c) prepare an automatic evaluation;

d) define the correlation mode for request-document correlation as the cosine mode;

e) choose the cutoff for request-document correlation as 0.35;

f) print out document and request vectors during the run;

g) weight concepts derived from statistical phrases twice as heavily as concepts derived from word stems;

h) end the specification list with the X specification.

*FILE ABSTR THES

The file to be read from the document tape is named ABSTR THES. At input time to MACTAP, this tape looks as follows:

*NAME ABSTR THES

*FIND QA1INFORM

binary looked up deck for query QA1INFORM

*FIND QA2

binary deck for query QA2 ...
...
...

*FIND QB16

a total of 35 questions are placed on the tape

*TEXT 01X3304 ...

binary deck for the first text, punched when looked up

...

*TEXT 82X1106 ...

binary deck for the last text

*STOP

*RELS

```
QA1INFORM    10
55XRELEVANT
23XRELEVANT
 ...
```

The relevant documents for the first question are listed

QA2

the request name for the second request and then its relevant documents are submitted, and so forth.

...

*RELS

*NAME ABSTR NULL

the next set of documents for MACTAP is provided here, after all relevant documents are introduced for the requests in the first collection.

*STOP

This card appears on A2. It indicates that no documents were submitted

on A2, only the set on A6 is to be processed.

end of file

This marks the end of the job for the monitor system. Total cards submitted: BCD 12, binary 2, total 14. Most of the production runs made with SMART are of this size.

8. Miscellaneous

SMART is frequently changed and revised; thus writeups may differ. The current writeup supersedes the ones included in reports ISR-7 and ISR-8 completely, and it supersedes report ISR-9 wherever the present writeup differs from the one in ISR-9. SMART is written in Fortran II and in FAP. With all its subroutines, the programs include approximately 30,000 source cards and 5,000 binary cards.

8.1. Size Limits

The thesaurus may consist of any number of English stems; however, only 32,000 significant concepts are allowed.

The program can address up to 250,000 documents; however, the capacity of the intermediate tapes will be exceeded before this point. Assuming that MACTAP is used to prepare the input document tape, about 25,000-100,000 documents (depending on their length) is a probable practical limit. If one insists on submitting documents on cards, about 1500-2000 documents is all will fit. The only truly significant limitation may be expected to be the fact that only fifty requests can be processed in one computer run.

8.2. Timing

The following timing estimates are approximations derived from our

experience on a batch-processing 7094 I. Since SMART is an experimental system, no great effort was spent on optimizing the object code for speed, and considerable improvements could be made in many of the programs.

Starting: Mounting two tapes, signing on, reading the specifications, etc. requires about two minutes. This represents mostly tape mounting time.

Lookup: To look up p words in a dictionary of q stems takes roughly $pq \cdot 10^{-7}$ minutes. Statistical phrase searching of p words in a list of q phrases takes about $\frac{1}{2}pq \cdot 10^{-5}$ minutes. Syntactic timing is exceedingly irregular with the old syntactic programs, and is effectively so slow that nothing useful can be accomplished in a reasonable amount of time. It is hoped that some syntactic analysis runs can be performed with a new revised analyzer to be distributed shortly.

Correlations: To correlate p requests against q documents and then sort the correlations, and evaluate, on the order of $\frac{1}{4}pq \cdot 10^{-3}$ minutes are used up. Present experimental data exist only for the range of p between 10 and 50, and q between 50 and 400; these estimates should not be trusted far outside this range.

Concept-concept correlations: If p concepts are involved (i.e. p = CONMAX - CONMIN) the first correlation takes about $\frac{1}{6}p^2 \cdot 10^{-4}$ minutes. Further iterations should be fast, assuming reasonable cutoffs.

Hierarchy: Most of the time is spent in tape shuffling, requiring about five minutes for collections of about 50,000 words.

9. Acknowledgments

The programs described here were written by Mark Cane, Tom Evslin, Guy Hochgesang, Alan Lemmon, Michael Razar, George Shapiro, and the author.

Other members of the SMART project have been Margaret Engel, Claudine Harris, Richard LeSchack, Edward Nelson, Joseph Rocchio and Edward Sussenguth. This project was supported by the National Science Foundation under grants GN-82, GN-360, and GN-495. The assistance of Prof. Susumu Kuno's research group, Dr. Owen Gingerich of the Smithsonian Astrophysical Observatory, and the operations staff of the Harvard Computing Center is gratefully acknowledged.

References

[1] Information Storage and Retrieval, Report No. ISR-9 to the National Science Foundation, Harvard Computation Laboratory, August 1965.

[2] G. Salton and M. E. Lesk, The SMART Automatic Document Retrieval System, Communications of the ACM, Vol. 8, No. 6, June 1965.

[3] G. Salton, Progress in Automatic Information Retrieval, IEEE Spectrum, Vol. 2, No. 8, August 1965.

[4] G. Salton, The Evaluation of Automatic Retrieval Procedures -- Selected Test Results Using the SMART System, American Documentation, Vol. 16, No. 3, July 1965.

[5] M. E. Lesk, The SMART System -- Typical Processing Sequences, Report No. ISR-8 to the National Science Foundation, Section I, Harvard Computation Laboratory, December 1964.

[6] M. E. Lesk, The SMART Automatic Text Processing and Document Retrieval System, Report No. ISR-8 to the National Science Foundation, Section II, Harvard Computation Laboratory, December 1964.

[7] Mathematical Linguistics and Automatic Translation, Report No. NSF-8 to the National Science Foundation, Harvard Computation Laboratory, 1962.

[8] Mathematical Linguistics and Automatic Translation, Report No. NSF-9 to the National Science Foundation, Harvard Computation Laboratory, May 1963.

[9] Mathematical Linguistics and Automatic Translation, Report No. NSF-13 to the National Science Foundation, Harvard Computation Laboratory, 1964.

[10] S. Kuno, Current Research in Computational and Mathematical Linguistics at the Computation Laboratory of Harvard University, Report No. NSF-15 to the National Science Foundation, Section I, Harvard Computation Laboratory, August 1965.

[11] M. Cane, Dictionary Lookup and Updating Procedures, Report No. ISR-7, Section IV, and Report No. ISR-9, Section V to the National Science Foundation, Harvard Computation Laboratory, June 1964 and August 1965.

[12] M. Cane, op. cit.

References (continued)

[13] C. Harris, Dictionary and Hierarchy Construction, Report No. ISR-7, to the National Science Foundation, Section III, Harvard Computation Laboratory, June 1964.

[14] J. Prowse, Syntactic Analysis of Incomplete Sentences in the SMART System, Report No. ISR-9 to the National Science Foundation, Section XI, Harvard Computation Laboratory, August 1965.

[15] M. E. Lesk, Evaluation of Retrieval Results in the Extended SMART System, Report No. ISR-9, to the National Science Foundation, Section XVII, Harvard Computation Laboratory, August 1965.

[16] G. Hochgesang, SOCCER - A Concordance Program, Report No. ISR-11 to the National Science Foundation, Section III, Cornell University, June 1966.