System Design Considerations
for Automatic Abstracting

A Thesis
Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Science.

By
Antonio Zamora, B.S.
The Ohio State University
1969

Approved by

[Signature]
Adviser
Department of Computer and Information Science
Acknowledgment

Thanks are due to Dr. James E. Rush, Professor of Computer and Information Science at the Ohio State University for evaluating the progress and results of this research; the author is also indebted to Mr. Richard Salvador, who worked closely with the author in the implementation of many of the concepts presented here for the IBM 1620 and IBM 360 computers.

Acknowledgment is given to the National Science Foundation for partial support of this work through a grant (GN 534) to the Computer and Information Science Research Center, and to the Data Center of the College of Commerce and Administration (James A. Black, Director) for allowing extensive use of an IBM 1620 computer during the initial phase of this work.

The author also wishes to acknowledge the support and encouragement given him by the Chemical Abstracts Service by whom he was employed during the course of this work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>I</th>
<th>Abstract</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>II</td>
<td>Statement of the Problem</td>
<td>vi</td>
</tr>
<tr>
<td>III</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>A. Purpose of Automatic Abstracting</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>B. Definition of an Abstract</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C. Previous Work in Automatic Abstracting</td>
<td>2</td>
</tr>
<tr>
<td>IV</td>
<td>System requirements for Automatic Abstracting</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>A. Methods of Sentence Selection</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1. The Title Method</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2. The Location Method</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>a. Contextual Inference</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>b. Rejection Criteria</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>c. Punctuation</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3. The Cue Method</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>a. Analysis of sentence weights</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>4. Frequency Criteria (the Key Method)</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>5. Editorial Clues</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>B. Intersentence References</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>C. Extension of Automatic Abstracting to other Languages</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>D. Limitations of Automatic Abstracting</td>
<td>20</td>
</tr>
</tbody>
</table>
Abstract

This thesis discusses the design of an automatic abstracting system. Methods for selecting material to create an abstract and a data structure which permits effective manipulation of the data are studied. Abstracts are produced solely by applying coherence and contextual inference criteria in the selection and rejection of sentences from the original document. Abstracts which are 10 to 20 percent the size of the original documents have been produced with the aid of a dictionary of approximately 700 entries.
Statement of the Problem

The initial work in automatic abstracting made use of statistical techniques; subsequent work in automatic abstracting led to the development of positional, cue word, and editorial criteria for selecting sentences for an abstract.

This thesis studies automatic abstracting techniques and their limitations. Existing methods for producing abstracts are extended and new approaches are suggested. The data structure and the computer programs used to test these methods are described.
INTRODUCTION

This paper discusses system design considerations that emerged during the development of computer programs to produce abstracts from full text and articulated indexes from abstracts or full text.

PURPOSE OF AUTOMATIC ABSTRACTING. Abstracts save time for the literature searcher by indicating whether the original document is relevant for a desired purpose or by providing the essential content of the original document. Clearly, the form and content of the abstract depends on its intended use.

Automatic abstracting (or automatic extracting) has as its goal the production of a condensed form of an original document by a computer; hopefully this will reduce the cost of producing abstracts and will make their availability more timely. Automatic abstracting is not limited to the selection of sentences as used by the author; it also has the function of modifying the sentences selected to produce a coherent and informative document. The function of abstracts, reviews of previous work in automatic abstracting, as well as detailed definitions of the objectives of automatic abstracting have been well documented by Wyllys (11), Edmundson (4), and Salvador (9).

DEFINITION OF AN ABSTRACT. An abstract, by definition, should be smaller than the original document; the content and length of an abstract depends on its intended use and on the type of abstract desired.

In this work an abstract is defined as a subset of the original document. The abstract consists of sentences contained in the
original document or of sentences derived from sentences in the original document. Thus, the abstracts produced by the programs discussed here consist of sentences selected from the original document, but some of the selected sentences are modified by removing certain of their parts. No attempt has been made to define the length an abstract should have. The abstract should include, though, the purpose of the work, and the conclusions of the author. Opinions, references to previous work, and subjective notions should be omitted because they are of low information value.

**PREVIOUS WORK IN AUTOMATIC ABSTRACTING.** The approaches taken by other workers in the preparation of automatic abstracts have been many and varied. Only those which are relevant to the subject of this thesis are presented here.

H. P. Luhn is given credit for first suggesting the possibility that a computer program could produce suitable abstracts (8). The general technique suggested by Luhn was to identify the most significant sentences of a document. Significance was to be judged by the representativeness of the words in the sentence, and the relative position of the representative words of a sentence.

Representativeness of the words was to be judged as follows: a) function words such as pronouns, prepositions, etc., were considered to have no value, b) the least frequent content words were not considered representative, c) content words with frequencies above the least frequent words were considered representative.

The proximity of words of high representativeness was used to
assign the sentence a value; a predetermined number of sentences with
the highest values were then printed in the order of appearance in the
original article. Luhn's approach is of importance because it has
been investigated by many other workers.

The Bunker-Ramo Corporation (previously called Thomson-Ramo-
Wooldridge, Inc.) has conducted extensive studies directed by H. P.
Edmundson on methods for producing automatic abstracts (2, 10). The
Bunker-Ramo studies have shown that there is great potential in four
methods of sentence selection; these are called the Cue, Key, Title,
and Location methods.

The Cue method makes use of a list of words which are classified
as: Bonus words (those that have a positive value or weight in sen-
tence selection), Stigma words (negative words), and Null words (those
which are irrelevant toward sentence selection).

The Key method is based on the frequency of occurrence of words,
and is similar in approach to Luhn's study.

The Title method is based on a glossary of the words of the title
and subtitles (excluding null words); sentences containing words co-
occurring in the title are assigned a higher weight than words co-
occurring in a subtitle.

The Location method is based on the hypothesis that certain
headings precede important passages and that topic sentences occur
early or late in a document or paragraph; this hypothesis was also
lent support by the observations of Baxendale in her studies of
indexing (1).
Edmundson recently evaluated these methods (3) and the results of his evaluation are in agreement with the results initially reported (2). It turns out that the Cue, Title, and Location methods improve the process of selecting sentences (individually or in combination), but the Key method actually degrades the selection process.

Edmundson and Wyllys (4) have suggested that frequency techniques should be based on the information-theoretic notion that the information value of a word is inversely proportional to the relative frequency of the word. This concept is diametrically opposed to Luhn's original suggestion, but it has a more appealing theoretical foundation. The implementation of this suggestion as it was postulated is relatively difficult because it requires complete statistical knowledge of all words in the language to be able to relate a word's relative frequency in general use to its relative frequency within a document.

Additional criteria, such as editorial clues, which can be used in automatic abstracting have not yielded consistently good results to warrant discussion here; for coverage of some aspects of these criteria see references 4, 9, and 11.
SYSTEM REQUIREMENTS FOR AUTOMATIC ABSTRACTING

For efficient processing, a language processing program should consider the largest independent item in its data base as its basic unit. Thus, in automatic abstracting the basic unit is an original article. Any approach which considers paragraphs or sentences as basic units is inadequate because there is interdependence between these moieties and the remainder of an article; a program which operates on interdependent units bears the burden of carrying information from one unit to the next.

Any automatic language processing program must have at its disposal the properties of the linguistic units, whether these be words, phrases, or sentences. Such requirements generally demand access to extensive dictionaries during processing; dictionary look-up tends to be a time-consuming operation which increases operating costs and running time significantly.

Some of the methods that can be used in automatic abstracting will be discussed in this section. Particular attention will be paid to the demands of each method on the system, and this information will be used in developing a general system design that complies with all the demands in an efficient way.

METHODS OF SENTENCE SELECTION. It is necessary to analyze the conditions under which the different methods of sentence selection are successful in order to develop criteria for selecting sentences to produce an abstract. Since an abstract can also be produced by rejecting sentences which are irrelevant, methods for rejecting sen-
sentences also deserve intensive analysis.

**THE TITLE METHOD.** The Title method has as a premise that the author describes in as few words as possible the essence of his paper; it can be assumed, then, that the words of the title are well chosen and consequently of high significance. With the use of a dictionary it is possible to eliminate most words and word combinations which serve as the grammatical framework for expressing the concept contained in the title (this aspect will be pursued further under the Cue method (p. 11)). Once the "framework" words are removed, the information-carrying words of the title have been isolated.

In the process of sentence selection, the words of each sentence should be matched against the information-carrying words of the title; if any words match, the sentence becomes a candidate for selection. The possibility should also be considered that the topic of the title (using words contained in the title) is likely to appear frequently, and if considerable reduction is desired additional criteria should be used before any sentence containing words co-occurring in the title is accepted for inclusion in the abstract.

**THE LOCATION METHOD.** The Location method is based on the physical arrangement of the linguistic elements of an article. This arrangement can be expressed in terms of two generalized descriptions: 1) the location of a sentence with respect to the limits of an article, and 2) the location of the phrases or words with respect to the limits of a sentence.

The location of a sentence with respect to the limits of an
article is governed by the style of the author or the editor; general writing guides provide advice about the placement of sentences within an article and suggest, for example, the use of introductory or summary sentences at the beginning and at the end of paragraphs. Of course, it is not possible to dictate in the matter of style and consequently the location of a sentence does not convey an unequivocal criterion for sentence selection or rejection.

The second description mentioned above is really a sentence description; the location of phrases and words within the sentence is subject to grammatical rules to which authors and editors adhere.

Even a partial syntactic analysis yields the basic sentence structure; this is possible because the basic sentence frameworks are limited in number. Punctuation clues are also helpful in determining sentence types. A question mark at the end of a sentence may indicate, for example, that the author is pondering about something, either because he does not know or as a way of introduction to his discussion. In any case, a question encountered in text (in organized text, that is) is likely to have been preceded by observations which precipitated the question and is likely to be followed by sentences which will try to explain the question raised by the author. The sentence that follows a question may well be something like:

"We will try to develop solutions to these questions in the following pages."

Contextual Inference. It should be noticed that the analysis of question marks as discussed above is concerned with contextual infer-
ences rather than with syntactic structure. Contextual inference is the basic concept behind sentence selection or rejection; it is on the basis of an inference that sentences containing words co-occurring in the title are considered for selection; in the case of questions, inferences are necessary to decide whether any sentence is to be rejected or selected for inclusion in the abstract.

It is advisable to reject questions from an abstract for two main reasons: 1) a question never provides related facts, and 2) if a question is selected, then the context which makes the question sensible also needs to be selected to preserve the coherence of the produced abstract (context preservation will be discussed further under Intersentence References (p. 17)).

Rejection Criteria. Before considering other punctuation clues, it is necessary to develop the implications of the reasons mentioned above for rejecting a particular type of sentence. Information content, information dependencies, and economy of expression are all involved. With these considerations in mind it will be fruitful to decide what an abstract should contain and what it should not contain.

An abstract should contain facts, the purpose of the paper and the conclusion of the author; an abstract should not contain opinions, references to previous work, questions, subjective notions, equations, tables, figures, references to equations, to figures or to tables. Depending on the application of the abstract, it might also be possible to exclude experimental setup, measurements, examples, and textual...
material which merely adds detail to the concept presented in the original paper. Specifying what an abstract should contain requires long lists of "desirable" concepts, whereas only a relatively short list of frequently occurring terms of low information value is needed to determine what should be omitted. Consequently, the process of rejecting sentences is much easier than the process of selecting sentences for an abstract.

Punctuation. The location of words or phrases with respect to the sentence provides a large amount of data for making contextual inferences. A "sentence" as discussed here refers to a string of words terminated by a period, question mark, or a semicolon. The question mark and the semicolon have a rather unequivocal use; the period, however, is used in abbreviations, ellipses and numbers as well as at the end of a sentence. These different usages need to be differentiated to delimit sentences properly.

Commas, like periods, can have several uses. Commas occur in numbers; they also separate items in series, parenthetical expressions, and clauses. Numerical and serial commas do not provide enough information by themselves to select or reject a sentence, but parenthetical commas generally indicate the use of synonyms, contrasts or comparisons, and dilatory or stylistic expressions.

While parenthetical expressions might not suffice to reject a sentence, the parenthetical expressions themselves can certainly be removed without affecting the total meaning of the sentence. Parenthetical expressions should definitely be removed when they are dil-
atory, stylistic, or when they contain a synonym, since in these cases nothing is lost. Removing parenthetical expressions which contrast or compare has the advantage of reducing context dependencies; the sentence:

"Our results, by contrast, show that X is blue."

can be successfully changed to:

"Our results show that X is blue.".

This second sentence conveys correct information regardless of what was said before, but this is not true of the first sentence. Thus, reduction of context dependencies produces more coherent abstracts.

Commas that separate clauses play a more vital role in sentence selection or rejection than other commas because they delimit the leading clause from clauses which qualify the leading clause. Keywords which indicate the relationship of the clauses are generally found adjacent to the commas that separate clauses; these keywords can be coordinating or subordinating conjunctions. Regardless of the type of keyword, second and subsequent clauses generally modify the first clause. The first clause, then is essential for the meaning of the sentence; if the first clause should be removed by the rejection criteria, the remainder of the sentence should also be dropped since it will not make sense otherwise. It is possible, on the other hand, to remove clauses other than the first and still obtain a sensible result; the sentence:

"The house was beautiful in the winter, but it was more comfortable in the summertime."

is reasonable when rewritten as:
"The house was beautiful in the winter."

Whether such transformations are used in the production of abstracts depends on the type of abstract desired; the loss of information produced by deletion needs to be evaluated against the brevity and the quality of the abstracts obtained.

Within clauses further reduction can take place by removing prepositional phrases; the loss of information resulting needs to be evaluated as above. Transforming the sentence:

"The house was beautiful in the winter."

to:

"The house was beautiful."

expresses the original idea but without all its qualifications.

**THE CUE METHOD.** It was mentioned earlier (p. 8) that opinions and subjective notions should not be contained in an abstract; with the application of those criteria it should be possible to reject the sentence:

"The house is beautiful."

on the grounds that "beautiful" is a subjective notion. Cue words, then, seem to have a more important role than location criteria in the rejection of sentences.

The Cue method provides a powerful approach to sentence selection or rejection. It was mentioned earlier that it was possible to decide what should be included in an abstract and what should be excluded; any words or combinations of words which are known to be used in stating the purpose of a paper, for example, should be coded to
possess a positive weight. "Our work", "This paper", etc., are certainly expressions which meet these criteria, but so is "This theme paper". It is necessary, then, to permit partial matches to allow for varied input while maintaining a limited list of cue words and expressions. A partial match occurs when one or more words intervene between any two words of an expression.

Opinions, references to figures, and other items which should not be included in abstracts can be identified by cue words such as "obvious", "believe", "Fig.", "Figure 1", "Table IV", etc. The weight of cue words can also depend on their position in a sentence; a sentence starting with "A" or "Some" is more likely to present detailed descriptions than a sentence which contains either of these words in a more central location of the sentence, because these words appearing at the beginning of the sentence have a strong quantitative function.

Cue words may also identify parenthetical expressions, idiomatic expressions, and cliches; cue words may also carry syntactic roles in cases where there is no ambiguity. The combination of all these properties makes it easier to determine algorithmically whether a sentence or phrase should be removed or retained.

Analysis of Sentence Weights. After weights have been assigned to the words of a sentence with the help of programs and dictionaries, it is necessary to decide whether that sentence is to become part of the abstract. Three alternatives exist:

1) evaluate the weight of the sentence by adding the weights of
the positive and negative words,

2) impose a hierarchy in which the weights for the words should be considered, or

3) a combination of 1 and 2.

The third alternative gives the best results because it is the most flexible. The first has the disadvantage of being blind to the data; if the sentence should contain the expressions "our work" and "unimportant", the sentence weight might be neutral and a significant sentence might be missed. The second alternative is too rigid and it might cause a sentence to be selected on the basis of a particular word combination even though there might also be several words with negative weights. Combining the hierarchy and weight methods yields flexible rules which form a hierarchy of qualified selection or rejection rules. An example of a rule in such a hierarchy might be: "If a cue word or expression of high value occurs in a sentence, select the sentence unless it also contains more than two low-value or more than four quantitative cue words.". This rule might very well precede a weaker selection rule or a rejection rule.

FREQUENCY CRITERIA (THE KEY METHOD). In the Introduction it was mentioned that the Key method of sentence selection, as suggested by Luhn, actually degraded the quality of the abstracts produced (2, 3). On the other hand, the technique suggested by Edmundson and Wyllis (4) of using information theory criteria would be difficult to implement because of the large volume of statistical information required.

Information theory criteria can be easily introduced into a
framework for abstracting like the one we have been discussing with successful results. It is necessary, however, to impose some limitations on how and at what point in the process of sentence selection or rejection the statistical criteria are applied. To illustrate this point let us consider three sentences:

1) "The detrimental nature of albinism has been well established."

2) "Albinism occurs at different frequencies in human populations."

3) "This paper discusses albinism caused by recessive autosomal genetic traits."

Sentence 1 is a likely candidate for rejection on the basis of the cue expression "well established"; sentence 2 does not have any cue words or expressions for selecting it or for rejecting it. Sentence 3 contains the cue expression "this paper" which makes it a candidate for selection. On the basis of cue words alone sentences 2 and 3 would be selected. Sentence 3 is chosen because of its positive value and 2 because there is no basis for rejecting it.

The expression "well established", which implies popular knowledge, makes sentence 1 a candidate for rejection regardless of the frequency with which the word "albinism" occurs in the paper being abstracted. The same is not true for sentence 2; if the word "albinism" is a high-frequency word, it may be possible to reject sentence 2 on frequency considerations. This approach still requires the word "albinism" to exist in a dictionary that specifies its fre-
quency of occurrence in natural language in order to have a basis for comparison. These considerations make it evident that cue words necessarily override frequency criteria, but it is also possible, in circumstances where the cue words have less than perfect reliability (e.g. homonymy), for frequency criteria to modify the weight of the cue words.

In the discussion of cue words the necessity of allowing partial matches was mentioned; this introduces an amount of uncertainty into the selection or rejection of sentences which can be reduced by introducing frequency considerations. The cue expression "this paper" would match against "this filter paper", "this wrapping paper" and other expressions which have nothing to say about the contents of the document being abstracted. Frequency criteria can be introduced as follows: if any cue expression exceeds a given frequency threshold, then its value should be reduced. This means that if the cue expression has a positive weight it should become less positive, and if it has a negative weight it should become less negative. With these guidelines it should be possible to produce abstracts of papers in which cue words are used in unusual ways. The thresholds at which the weight transitions should take place need to be determined, but statistical data is needed only for the cue expressions contained in the dictionary, rather than for the whole language.

Certain cues of negative value can not be handled as stated above. References to figures and graphs are examples of this. A reference to a figure is meaningless when there are no figures in the abstract.
In the case where frequency criteria indicate a large number of references to items which are not to be carried in an abstract, such as graphic material, the abstracting algorithms should remove such references and include concluding sentences such as: "Graphic material is presented."

**EDITORIAL CLUES.** Editorial criteria such as italics, capitalization, and section headings provide useful clues: Capital letters indicate proper names, the beginning of sentences, and acronyms; section headings indicate the topic of the next paragraphs; italics are used for foreign language quotations, names of bacteria, etc. The use of italics is not as standardized as the use of capital letters; thus, italics are less useful to an algorithm than the clues provided by capital letters.

If editorial clues are to be used in abstracting, these clues must be incorporated into the machine representation of the data; unfortunately, if the data representation differs, the machine will consider capitalized words different from the same words in lower case letters. The solution to this problem is to include such features separated from the data and have the data in a common representation, all lower case, for example. Every word or expression would then have a series of properties which describe it; typical properties would be:

1) location of the word within the machine,
2) length of the word,
3) type of print and capitalization attributes of the word,
4) syntactic role of the word,
5) weight of the word with respect to abstracting,
6) context dependency properties,
7) alphabetic rank in the document, and
8) relative frequency of the word in the document.

**INTERSENTENCE REFERENCES.** Intersentence references give much information about the logical relationships within the text material, but they require involved treatment if a coherent abstract is to be produced. In an earlier discussion (p. 10) it was pointed out that if more than one clause exists in a sentence then the first clause is indispensable to the meaning of the sentence, i.e., it carries more information than subsequent clauses. Generally the first clause will also contain intersentence references if there are any. If there are words which require antecedents in the second and subsequent clauses they generally refer to the first clause. Some cue words that indicate intersentence references are: "these", "they", "it", and "above". When these words have multiple uses, additional criteria are required for determining if there is intersentence reference. The expression "It is known that ... " does not refer to a previous sentence, but "It was spinning slowly" refers to a previously named object. Noticing patterns in the use of words such as "it" makes possible the use of these words to detect intersentence references. The following rule, for example, contends with many exceptions in the use of the word "it": "It" in the first or only clause indicates intersentence reference unless it is followed closely by "that".

A word like "above" confronts us with a different challenge—
of enumeration.

"The use of water was mentioned above."

and

"Water evaporates rapidly above 80 degrees centigrade."

are two sentences which make use of different meanings of the word "above". By enumerating those cases that we are interested in, we can determine intersentence references. Thus, the cue expressions "presented above", "mentioned above", and "stated above" would detect intersentence references.

We conclude, then, that cue words to determine intersentence references can be made to represent rules ("it ... that", for example), or they can enumerate the items of interest.

There are intersentence references that do not make use of any cue words; instead, they use the name of the antecedent rather than a pronoun. Consider the sentences:

"Substance X and substance Y form solutions in ammonia.

The solutions are blue."

Intersentence references such as these can be detected in a manner similar to the Title method of sentence selection. If any non-function words co-occur in adjacent sentences they are likely to be closely related.

The processes of sentence selection and rejection must take into consideration the coherence of the abstract. Suppose that a sentence that requires an antecedent is to be included in an abstract; it is necessary to check if the previous sentence has been removed and to
reinstate it if necessary. If the restored sentence also requires an antecedent the procedure must be repeated. It can be decided that if many sentences would be reinstated because of the required antecedents of one sentence, then it is not worthwhile keeping that one sentence.

It is important also to consider semantic interrelationships when there are intersentence references. Suppose that a sentence that requires an antecedent does not have any positive or any negative weights. We can decide that if the previous sentence was deleted because of a negative weight then this sentence should also be removed, but if the previous sentence was selected then the sentence under consideration should also be selected. When a sentence that requires an antecedent has a negative weight it can be removed without affecting the coherence of the abstract.

EXTENSION OF AUTOMATIC ABSTRACTING METHODS TO OTHER LANGUAGES. As indicated on page 16, all the material which needs to be used for preparing an abstract can be contained in the properties which describe the words. A program which produces abstracts by referring to the properties of the words rather than the words themselves, produces the same results whether the words are French, Spanish, or English. Differences in processing the different languages would be required only when the grammar of the languages is markedly different; English, Spanish, and other romance languages have prepositional structure; they also have similar punctuation conventions, etc. Differences such as location of adjectives with respect to the
nouns is not important since these features are not used in sentence selection; they may become important if prepositional phrases are to be removed, though. The cue word dictionary can be the source of syntactic, semantic, and frequency information. Thus, just a single pass against the dictionary is needed to supply the data-dependent information. An internal sorting program can be used to supply alphabetic rank keys to be used for matching against the dictionary, and syntactic programs can supply information on the type of periods, commas, and phrases encountered in the document. It can be seen, then, that the syntactic programs complement the information supplied by the dictionary. Once all this data has been amassed, phrase and sentence selection can proceed without referring to the text again except for printing the selected material.

**LIMITATIONS OF AUTOMATIC ABSTRACTING.** Sentence and vocabulary analysis are two problems encountered in automatic abstracting for which many solutions have been formulated by workers in computational linguistics. Many of these solutions are not particularly suited to automatic abstracting, but the knowledge is already available; all that is needed is to adapt that knowledge to a particular application. Thus, sentence and vocabulary analysis are not permanent limitations in automatic abstracting.

The organization and style of an article, as well as its content, impose limitations for which suitable solutions are not yet available. The organization of an original article is important because the sentences selected for the abstract are generally placed in the order in
which they occur in the original. This need not be so; earlier, we considered introducing sentences which represent particular characteristics of an original such as "Graphic material is presented.". Similarly, elimination or proper manipulation of intersentence references might yield coherent abstracts. Actual reorganization of the sentences selected for the abstract might prove beneficial only occasionally; machine implementation of text reorganization may be quite difficult and perhaps impractical because of its limited application.

Technical reports are principally expository in style; automatic abstracting is generally directed to process this style. Other styles present problems which are not so readily tractable. It is not expected that dialogues, poetry, or literary works would need to be abstracted since they are more suited for critical or interpretative reviews; they are clearly outside the domain of automatic abstracting.

Articles with imbedded languages such as mathematics or foreign languages are hard to deal with. Such articles can not be abstracted readily without a knowledge of the semantics of the languages involved and without a way to distinguish them. If a mathematical formula, for example, contained the letter "A" as a variable symbol, the abstracting program would need to distinguish it from the article "A" to be able to properly define intersentence references and syntactic structures. Furthermore, the program would have to have sufficient information to decide whether an equation should be selected for the abstract or not. It is not clear at this point that any of the se-
lection clues mentioned above would be suitable for such an undertaking, but certainly much more information will be needed in the selection process. Editorial clues are likely to be of great help in this area.
PROGRAM REQUIREMENTS

Automatic language processing involves manipulation of variable length strings (words) which possess syntactic and semantic attributes. An automatic language processing system must solve the problems of organizing the data and defining the program interaction. The organization of the data should be such that the strings of the language and their attributes are made readily available to the programs, and it should be flexible enough to permit reorganization of the data. The program interaction should be minimized by restricting communication through a common interface; this allows programs to be changed or improved independently of the rest of the system.

Automatic abstracting has data manipulation requirements which allow the use of a specialized data organization with great effectiveness. This section discusses the data organization requirements and the program interactions of automatic abstracting.

The ideas presented here have been implemented on the IBM 1620 and IBM 360 computers.

DATA ORGANIZATION. In automatic abstracting it is necessary to be able to identify words and expressions and to attach to them attributes derived by programs or obtained from dictionaries. The way in which the data is to be manipulated is a very important consideration in defining the data organization. Some of the manipulations include matching words against dictionaries to identify multi-word expressions, matching words against other words in the article, deletion of words, and scanning for word attributes to the right or to
the left of any given word.

Two data structures satisfy these requirements: lists and tables.

LISTS. A simple list consists of a pointer to the data and a pointer to the next element in the list. It is easy to add or to delete elements by manipulating the pointers of the list, but they have some notable disadvantages. With a simple list it is not possible to scan backward, since the pointers point to the next item only. A symmetric list, on the other hand, has a pointer to the data, a pointer to the preceding element and a third pointer to the next element in the list; this type of list can be scanned backward.

A member of a list can be accessed only by starting at the first member of the list (or also at the last member of a symmetric list) and following the pointers until the member is found; for this reason strategies to reduce search time can not be employed.

Lists require a large amount of storage. A simple list requires two pointers for every word of data, and a symmetric list requires three pointers; these pointers and the programs to manipulate them can require up to three times the amount of storage required by the data.

TABLES. Tables consist of a set of arguments each of which is associated with a unique key. The key may be explicit or implied. Information is retrieved from a table by locating the desired key and obtaining the arguments corresponding to the key; the arguments may be updated in a similar manner.

Tables can store information in a form well suited for retrieval,
although their structure tends to be rigid. Ordered tables consisting of entries that are all equal in length can be scanned forward or backward and techniques such as binary searches and address calculation, as well as sequential searches, can be performed to minimize search time. Ordered tables have the disadvantage that additions require moving many entries to make room for the new entry. Another way of adding an entry to an ordered table is to tag the entry which would precede it and add a pointer to the new entry. This solution has many of the disadvantages of lists. In unordered tables new entries can be added at any available location, but only sequential searching or hashing techniques can be used for retrieving information.

Tables of fixed length entries are generally represented by vectors within a computer. A vector is a set of elements which are physically adjacent in the computer memory. A vector is defined by its base address, element size, and its length.

Data Structure for Automatic Abstracting. The data structure for the implementation described herein incorporates some features of lists (pointers to the data) and some features of tables (storage of word attributes). Three constituents comprise the data structure; they are:

1) A workarea, where the text is stored throughout processing.

2) An attribute vector, which contains pointers to each word of the text, attributes such as length, and semantic and syntactic attributes for the corresponding words. Textual properties such as capitalization could also be incorporated
in the attribute vector. The nth element of the attribute vector corresponds to the nth word of the text in the workarea.

3) An alphabetic vector, which defines the alphabetic rank of the words of the text. The nth element of the alphabetic vector contains the number of the attribute vector element which corresponds to the nth word in alphabetic sequence. The alphabetic vector permits matching against a dictionary without reorganizing the data.

The attribute and alphabetic vectors are combined to form a table; it is possible to re-use the space of the alphabetic vector after all alphabetic processing has taken place, hence combining the vectors results in space savings. Figure 1 shows the constituents of the data structure. Notice, for instance, that the first element of the alphabetic vector in Figure 1 points to the sixth element of the attribute vector. The sixth attribute vector element, in turn, corresponds to the first word in alphabetic sequence.

Logical Operations Possible. The workarea and table structure described above have properties which are suitable for automatic abstracting and for text searching. Matching words against a dictionary in alphabetic sequence can be done by referring to the alphabetic vector; the arguments of a match can be stored in the attribute vector. Information may also be entered into the attribute vector at the time at which the vector is constructed to identify punctuation, textual, and editorial properties of the data. The table can be scanned backward
CONTENTS OF WORKAREA

MACHINE ADDRESS: 0 4 10 14 18 23 31 39 46

TEXT: The rocks did not have sharply angular corners.

TABLE CORRESPONDING TO THE WORKAREA

<table>
<thead>
<tr>
<th>implied vector element numbers</th>
<th>Attribute vector</th>
<th>Alphabetic vector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>word</td>
<td>word</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>31</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>39</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>46</td>
</tr>
</tbody>
</table>

DATA STRUCTURE FOR AUTOMATIC ABSTRACTING

FIGURE 1
or forward to facilitate processing.

Deletion of words can be effected by introducing "deletion" attributes to permit temporary or permanent deletion of words. Deleted entries are ignored by all the programs of the system. It is desirable to have both permanent and temporary deletion attributes to simplify some of the processing programs.

Substitution of words simply requires updating the length and address in the appropriate table entry; the words themselves are not moved.

Insertions can not be conveniently made in the table structure; it is possible, however, to substitute a multi-word expression for a single word to achieve the effect of an insertion. When this is done, the set of attributes in the table entry applies to the group of words, rather than to the words individually. However, the difficulty of inserting words presents no problem in automatic abstracting since this procedure is not likely to be used to any great extent.

The table provides fast access to the data in the workarea; this is a useful property not only for automatic abstracting, but also for text searching. For text searching, it is better to have the displacements of words from the beginning of the workarea in the table, rather than their machine addresses; this permits relocation of the data in the machine memory. If it is necessary to find a particular word in the workarea, a binary search or an address calculation procedure may be used to obtain this information. In either case, an element of the alphabetic vector must be located to obtain the address
portion from the corresponding attribute vector element. On the basis of the comparison of the words it can be determined whether it is necessary to advance or to go back along the alphabetic vector.

In summary, the data is stored in the workarea in its original order and is made directly accessible by means of the table. This data structure serves as the interface through which all of the programs of the system interact.

**PROGRAM INTERACTION.** In a system with many programs, all of which operate on the same data, it is desirable to coordinate the program interaction through a common interface to allow individual programs to be modified or improved without affecting the whole system. The data structure for automatic abstracting described above contains all the information that any program of the system requires. Each program operates by modifying the data structure or by referring to the table. Every program, then, has access to the result of all programs which were executed before it; this allows any program to override or modify the results of previous programs. The only parameters which a program needs to perform its particular function are:

1) The address of the workarea,
2) The address of the table, and
3) The number of entries in the table.

The programs described in this section make use of subroutines which locate information or modify attributes in the table. The parameters of the subroutines generally are two table addresses plus information which depends on the function of the subroutine. The
two table addresses delimit the portion of the data structure on which the subroutine is to operate.

**SYSTEM IMPLEMENTATION.** The automatic abstracting system was originally programmed for an IBM 1620 computer (model I) with a capacity of 20,000 storage positions. The peripheral devices used were a card reader/card punch, and the console typewriter. The programs were written in IBM 1620 Symbolic Programming System (5); they required an overlay structure in this machine. The programs were subsequently converted for an IBM 360 model 75. The conversion required changes in the storage allocation for the data structure to make efficient use of the binary data representation of the IBM 360. A table entry of the data structure can be stored in 8 bytes in the IBM 360; an equivalent table entry requires 20 storage positions in the IBM 1620. The programs for the IBM 360 were written in assembler language (6, 7). Although the IBM 360 has a more powerful instruction set than the IBM 1620, the total number of instructions of the programs was not reduced because the IBM 360 does not use indirect addressing.

The programs written for the IBM 360 eliminated deficiencies which existed in some of the programs for the IBM 1620. Also, the new programs take advantage of the data management functions of the operating system.

Figure 2 illustrates the program interaction; the functions of the programs and subroutines are outlined below (the logic of these programs is explained under ALGORITHMS, p. 34):
Program Interaction
Figure 2
<table>
<thead>
<tr>
<th>Program name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARDDISK</td>
<td>Creates a dictionary from a card file.</td>
</tr>
<tr>
<td>MAIN</td>
<td>Reads an original article and constructs a table entry for every word and punctuation mark. Punctuation attributes are stored in the table at the time it is constructed. The program MAIN also coordinates the execution of all other programs.</td>
</tr>
<tr>
<td>SORT</td>
<td>Sorts the words of the article using the members of the alphabetic vector as keys. Table entries with punctuation attributes are ignored during the sorting procedure.</td>
</tr>
<tr>
<td>FREQ</td>
<td>Lists the words of the original article and their number of occurrences; the printed output is used to manually study the statistical properties of the data. This program is not used in producing abstracts.</td>
</tr>
<tr>
<td>WORDCTRL</td>
<td>Enters the information contained in the dictionary created by CARDDISK into the attribute vector of matching entries. Multi-word entries are matched before single word entries. Some attributes are altered depending on the number of occurrences of the dictionary entry in the original article (Frequency Criteria, p. 39).</td>
</tr>
<tr>
<td>SEMANTIC</td>
<td>Incorporates all the rules for producing an abstract from the original.</td>
</tr>
<tr>
<td>PRINT1</td>
<td>Prints the original or the abstract.</td>
</tr>
</tbody>
</table>
**INDEX** Creates an articulated index of the original article or of the abstract.

<table>
<thead>
<tr>
<th>Subroutine name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECKROL</td>
<td>Scans forward (left to right) along a section of the table to locate the attributes specified; it locates the first table entry which contains the desired attributes.</td>
</tr>
<tr>
<td>CHKROLEF</td>
<td>Has the same functions as CHECKROL, but the scan proceeds backward.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Places deletion attributes in entries of the table delimited by two addresses.</td>
</tr>
<tr>
<td>RESTORE</td>
<td>Nullifies deletion attributes in a section of the table delimited by two addresses.</td>
</tr>
<tr>
<td>CHCOMA</td>
<td>Classifies the non-numeric commas of a sentence as parenthetical, serial, or clause commas.</td>
</tr>
<tr>
<td>PERIOD</td>
<td>Identifies sentences. It scans the table until it locates a punctuation attribute of <em>semicolon</em>, or an attribute of <em>period</em> which is not part of an abbreviation. This subroutine also indicates when the complete article has been scanned.</td>
</tr>
<tr>
<td>RELEVANT</td>
<td>Determines relevance of sentences to the title and intersentence references where cue words are not used. This subroutine compares the words corresponding to table entries which have no attributes; an attribute</td>
</tr>
</tbody>
</table>
indicating relevance is inserted in the table entries of words that match.

ALGORITHMS. This section explains the logic of the programs that comprise the heart of the automatic abstracting system. CHOMA is the only subroutine that will be discussed here because the other subroutines have very simple functions. The programs of the present system can be improved by modifying the logic or altering their design; appropriate modifications will be discussed with each particular program. The program that produces the articulated index has been described by Salvador (9) and will not be discussed here. (Only the programs for the IBM 360 will be discussed because they are more general than the programs for the IBM 1620.)

CARDDISK. The CARDDISK program creates a dictionary on a direct access device from a card file. It is necessary to read the dictionary once for each original article to be abstracted, however, a dictionary of small size could be stored in the machine memory, thus reducing processing time.

The dictionary presently contains about 700 entries averaging approximately 10 characters per entry (see Appendix). This dictionary has allowed production of abstracts which were 10 to 20 percent the size of the original article. Because of the significant reductions achieved with this dictionary, it is anticipated that it will not increase markedly in size; thus, it would be possible to store the dictionary in the machine memory and a considerable gain in speed would result.
MAIN. This program reads the original article from punched cards into a workarea. All characters of the IBM 029 keypunch are accepted (Figure 3). The first non-abbreviation period followed by a blank indicates the end of the title. If there is a bibliographic reference, it is enclosed in pound signs at input and immediately follows the title. Bibliographic information, if present, is disregarded in abstracting but is retained for the abstract. An options card, identified by dollar signs in the first two columns, indicates the end of the article. This card also contains asterisks which indicate the output options desired for the article. The options presently available are:

1) Print the words in the document and their frequencies.
2) Print the original document.
3) Print an index of the original.
4) Print the abstract.
5) Print an index of the abstract.

The MAIN program creates the data structure for the abstracting algorithm. MAIN creates a table entry for every alphanumeric string delimited by blanks or by special characters; a table entry is also made for every special character with the following exceptions:

A hyphen is considered a special character only when it is preceded and followed by alphabetic characters or blanks. This allows generation of a single table entry for strings such as -175 and 1-bromo-2-naphthol. A string such as multi-valued, however, generates three table entries.
Alphabetic characters  A-Z

Numeric characters  0-9

Blank

Special characters  $ . ( + & ! $ * )
                  ; - / , % _ ? : #
                  @ ' = " < | > ~
Commas, periods, right parentheses, and the right caret ("greater than" sign) are required to have a blank to their right to be considered special characters. The left parenthesis and the left caret ("less than" sign) are required to have a blank to their left. Numeric commas and periods are not followed by blanks, hence are considered part of the numeric character string. The restriction imposed on the parentheses and carets applies to some technical writing styles.

MAIN stores the following information in the table:

1) Length and address of the word or special character.
2) Semantic and syntactic attributes of special characters (defined as the binary value of the character).
3) A sequential number in the alphabetic vector element (starting from zero for the first entry in the table).

The table entries constructed by MAIN occupy 8 bytes. Each table entry can contain the information listed below.

<table>
<thead>
<tr>
<th>Byte of Table entry</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Length of the word or special character</td>
</tr>
<tr>
<td>2 to 4</td>
<td>Address of the word or of the special character</td>
</tr>
<tr>
<td>5</td>
<td>Semantic attribute</td>
</tr>
<tr>
<td>6</td>
<td>Syntactic attribute</td>
</tr>
<tr>
<td>7 to 8</td>
<td>An alphabetic vector element is stored here during alphabetic processing. After all alphabetic processing is completed, the deletion attributes are stored here.</td>
</tr>
</tbody>
</table>
If the MAIN program were expanded to accept upper and lower case characters, it would be necessary to store data in a common storage mode (e.g., lower case), and include capitalization attributes in appropriate Table entries.

Once the program MAIN has created the Table, it coordinates the execution of other programs according to the options specified in the options card.

SORT. The sorting method used for the IBM 360 is an improved linear selection with exchange key sort method. The method takes advantage of the fact that all words are at least one character long, and that in a large number of the comparisons it is possible to determine the sorting sequence by comparing first characters. Thus, only when the first characters are equal is it necessary to obtain the lengths to make a complete comparison.

The keys used by this program are the alphabetic vector elements of the Table. Entries with special character attributes are ignored during sorting. The alphabetic vector after sorting is illustrated in Figure 1; notice that the nth alphabetic vector element contains the number of the table entry which corresponds to the nth word in alphabetic rank.

WORDCTRL. The WORDCTRL program reads the dictionary and assigns the attributes found there to the terms of the article that match those of the dictionary. Frequency criteria alter the attributes stored in the Table, and a hierarchy is used for assigning the attributes to multi-word terms that have words in common. In order to
apply frequency criteria and to match against multiple-word entries.

It is necessary during processing to go back to alphabetic entries
that have already been passed; the Table structure makes this a rel-
atively easy task.

The dictionary read by WORDCTRL is in alphabetic sequence by
the first or only word of the term; multi-word terms, however, precede
single word terms.

**Frequency criteria.** Frequency criteria are presently applied by
the WORDCTRL program in a very rigid manner; specific frequency thresh-
olds and specific attribute conversions are used for each term depending
on the attributes of the dictionary entry. It would, however, be
possible to specify the frequency threshold and the alternative attri-
butes to be used for each dictionary entry.

When a dictionary match is found, a dummy attribute is inserted
into the Table. The dictionary entry continues to be compared against
the words of the article (by using the alphabetic vector) until the
words of the article are higher in sequence than the dictionary entry,
or the end of the article is reached. At this point the number of
matches is used to compute whether the attributes of the dictionary
entry are to be modified. If the number of words in the article is
under 500, the actual number of matches is used in the computation; if
the number of words in the article exceeds 500 then the number of
matches per thousand words of the article is used.

Words with positive attributes are given less positive values if
the number of matches per thousand exceeds four; words with negative
attributes are made less negative if the number of matches per thousand exceeds seven. This evaluation scheme favors the decrease of positive values, and thus, it leans in the direction of smaller abstracts. When the new attributes are determined, they replace the dummy attributes stored in the Table earlier.

Multi-word entries. Matching against multi-word entries proceeds as follows: When the first word of a dictionary entry matches a word in the article, a pointer (KK1) is loaded with the address of the current alphabetic vector element. The word that follows the matching word is then compared against the second word of the dictionary entry; if the words are equal the process is repeated until all the words of the dictionary term have been compared. It should be noticed that the alphabetic vector element is used in locating the first word only; subsequent comparisons refer to adjacent Table entries. When all the words of multi-word terms match, the Table entry of the first word carries the attributes found in the dictionary (unless modified by frequency criteria) and the succeeding Table entries are assigned continuation attributes. Partial matches are allowed by permitting up to three table entries, none of which have punctuation attributes, to intervene between words of multi-word dictionary entries. For example, the dictionary entry THIS PAPER would match the words THIS THEME PAPER as well as the unqualified expression.

Dictionary entries are compared against the words of the article by using the alphabetic vector until the words of the article are higher in sequence or exhausted; after the dictionary entry is processed,
the next dictionary entry is read. The first word of the new dictionary entry may be equal to the first word of the previous dictionary entry; this makes it necessary to start comparing against the article at the point indicated by the pointer KK1.

**Hierarchy of attributes.** It is necessary to have a hierarchy for introducing attributes into the table when multi-word terms overlap. If the first words of the dictionary terms are identical, the hierarchy is built into the dictionary by ordering the terms according to the number of words they contain, the terms with most words coming first. Ordering the dictionary in this way allows the longest and most restrictive terms to match first.

When the overlapping words are not the first words of the term, the program must impose rules of precedence. Syntactic attributes, for instance, are overlaid by the attributes of succeeding terms so that the last term that matches dictates the syntactic attributes. Semantic attributes, by contrast, do not override previously assigned semantic attributes, although they will overlay continuation attributes.

**CHCOMA.** CHCOMA is used by the SEMANTIC program to permit processing of the clauses of a sentence. This subroutine classifies the commas of a sentence into categories which are recognized by the SEMANTIC program. The output of the CHCOMA subroutine is aimed at the specific requirements of the SEMANTIC program. The SEMANTIC program is concerned with three types of commas: serial, parenthetical, and clause commas. Serial commas are generally skipped, parenthetical commas delimit portions of a sentence which are to be deleted, and
clause commas identify the smallest coherent units of a sentence.

The algorithm used by CHCOMA is not fool-proof, but it contends with a large number of cases. The IBM 1620 algorithm processed one comma at a time; the algorithm used in the IBM 360 examines a complete sentence. The latter approach makes all the relevant information available to the algorithm and better results are obtained.

CHCOMA examines the syntactic and semantic attributes in the table and returns to the semantic program a list of pointers; each pointer identifies the table location of a comma and its type. If there are no commas, the list is empty. The following criteria are used by the subroutine CHCOMA:

1) If the comma is immediately followed by a conjunction and the previous comma (if any) was serial then this comma is serial.

2) If the comma is immediately preceded by a semantic attribute for parenthetical expressions, the comma is parenthetical; the previous comma (if any) is categorized as serial regardless of its previous categorization.

3) If the comma is followed by a pronoun, the comma is a clause comma.

4) If the comma is followed by "TO" or a verb, the comma is parenthetical and the previous comma is categorized as serial; however, if the comma is the first comma of the sentence, it is considered serial.

5) If none of the above conditions have been met, six table
entries following the present comma, or as many table entries as remain in the sentence are examined for the attributes *conjunction, comma, verb, and preposition* (excluding "Op"). The relative positions of these attributes determine the type of the comma.

5a) If none of the above attributes is found in the table entries following the comma, the comma is a clause comma.

5b) If a comma or conjunction is found, but no verb or preposition, the comma is serial.

5c) If a verb or preposition is found, but no comma or conjunction, the comma is a clause comma.

5d) If either a preposition or verb occurs before a conjunction or comma in the section of the table examined, the comma is a clause comma.

5e) If either a conjunction or comma occurs before a preposition or a verb, the comma is serial.

**SEMANTIC.** In the early stages of the design of these programs it was realized that a computer program provided a very rigid framework for implementing the sentence selection rules of automatic abstracting. This prompted us to separate, as much as possible, the sentence selection procedure from the words of the language. Thus, the dictionary was designed to contain the semantic and the syntactic attributes of the words, and the abstracting program was designed to operate solely on the attributes of the words. Of course, since such attributes cannot be unambiguously assigned by a simple matching procedure, programs
such as CHICOMA and WORDCTRL supplement the information obtained from the dictionary. Separating the abstracting algorithm from the words of the language has two principal advantages:

1) The algorithm may be applied to more than one natural language by using different dictionaries.

2) Abstracts with different orientations can be produced by manipulating the attributes of the dictionary terms.

The second point makes it very easy to study the impact of different classification schemes for terms, without having to resort to program modifications. If the abstracting program is flexible enough to allow a large number of logical operations, the main problem in the production of abstracts becomes the classification of the dictionary terms.

The abstracting program described here consists of a hierarchy of alternating positive and negative selection rules. Each selection rule may reference one or more semantic or syntactic attributes and may affect whole sentences or parts of sentences. Alternation of selection and rejection criteria allows the first rule which applies to any sentence or part of a sentence to determine whether the item under consideration is to be removed or retained.

The Dictionary. Dictionary terms contain two attributes: 1) a syntactic attribute, and 2) a semantic attribute.
The syntactic attributes have the following codes and meanings:

<table>
<thead>
<tr>
<th>Syntactic code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Article</td>
</tr>
<tr>
<td>C</td>
<td>Conjunction</td>
</tr>
<tr>
<td>D</td>
<td>Deleted word</td>
</tr>
<tr>
<td>F</td>
<td>Null word</td>
</tr>
<tr>
<td>J</td>
<td>Continuation of a previous syntactic attribute</td>
</tr>
<tr>
<td>N</td>
<td>Pronoun</td>
</tr>
<tr>
<td>P</td>
<td>Preposition</td>
</tr>
<tr>
<td>O</td>
<td>Exclusively assigned to &quot;OF&quot;</td>
</tr>
<tr>
<td>Q</td>
<td>Exclusively assigned to &quot;TO&quot;</td>
</tr>
<tr>
<td>R</td>
<td>Exclusively assigned to &quot;AS&quot;</td>
</tr>
<tr>
<td>S</td>
<td>Subject heading (used by the INDEX program)</td>
</tr>
<tr>
<td>V</td>
<td>Verb</td>
</tr>
<tr>
<td>W</td>
<td>Auxiliary verb</td>
</tr>
<tr>
<td>X</td>
<td>Exclusively assigned to &quot;IS&quot;, &quot;ARE&quot;, &quot;WAS&quot;, and &quot;WERE&quot;</td>
</tr>
<tr>
<td>Z</td>
<td>Negative</td>
</tr>
</tbody>
</table>

Notice that some syntactic attributes are assigned to specific words or groups of words which play special grammatical roles; this allows proper operation of the programs that handle syntactic information.
The semantic attributes have the following codes and meanings:

<table>
<thead>
<tr>
<th>Semantic code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Assigned to very negative terms; those which do not belong in an abstract (e.g., obvious, interesting)</td>
</tr>
<tr>
<td>B</td>
<td>Parenthetical expressions, terms of low information content, or terms which are associated with items of low information content (e.g., however)</td>
</tr>
<tr>
<td>C</td>
<td>Used for words which require an antecedent (e.g., this, these)</td>
</tr>
<tr>
<td>D</td>
<td>Deleted word (e.g., very)</td>
</tr>
<tr>
<td>E</td>
<td>Used for quantifiers (e.g., many, more)</td>
</tr>
<tr>
<td>F</td>
<td>Null (assigned to abbreviations)</td>
</tr>
<tr>
<td>G</td>
<td>Assigned by the program to indicate relevance between sentences or relevance to the title.</td>
</tr>
<tr>
<td>H</td>
<td>Terms which introduce modifying phrases (e.g., whose)</td>
</tr>
<tr>
<td>I</td>
<td>Used for very positive terms; those which almost unequivocally are related to something of importance (e.g., our work)</td>
</tr>
<tr>
<td>J</td>
<td>Continuation of a previous semantic code</td>
</tr>
<tr>
<td>K</td>
<td>Assigned to terms which are related to items of high information content (e.g., important)</td>
</tr>
<tr>
<td>L</td>
<td>Introductory qualifiers (e.g., once, a)</td>
</tr>
</tbody>
</table>

The Abstracting Algorithm. The abstracting algorithm is presented as a series of rules written in the form of PL/1-like statements when made necessary by the complexity of the logic. If a rule does not
apply, the next rule in the hierarchy is applied. It should be kept in mind that when the SEMANTIC program is executed, the WORDCTRL program has already assigned attributes to terms which match against the dictionary.

The first three rules are applied only when beginning the processing of the article.

RULE 1: Scan to the first non-abbreviation period; assume this to be the end of the title.

RULE 2: The bibliographic reference, if any, is enclosed within pound signs and immediately follows the title; it is skipped if it is present.

RULE 3: The title is matched against the complete article using the RELEVANT subroutine. This introduces the semantic attribute $G$ into table entries of words which have no attributes and which match a word of the title.

All the following rules are applied to every sentence of the article (a sentence is obtained by using the PERIOD subroutine).

RULE 4: The semantic attribute $I$ indicates importance. A sentence which contains an $I$ attribute is retained if it meets coherence criteria. The rule is the following:

\[
\text{if (non-negative I in sentence)} \quad \text{then}^1 \\
\text{remove parenthetical expressions;}
\]

\[
\text{/* the coherence criteria comprise the next set of if statements */}
\]

\[
\text{if (C in first clause)} \quad \text{then}^2
\]
if (previous sentence gone) then

/* important sentence that requires an antecedent and the previous sentence is gone */

restore up to 3 sentences;

if (verb in sentence) then
sentence stays;
else
delete sentence and those restored;

else

/* important sentence that requires an antecedent and the previous sentence stayed */

if (verb in sentence) then
sentence stays;
else
delete sentence;

else

/* important sentence that does not require an antecedent */

if (verb in sentence) then
sentence stays;
else
delete sentence;

else

Notes:

a) The superscripts identify corresponding alternatives.

b) A "non-negative I" is defined as a table entry which contains a semantic attribute of I and is preceded by two table entries neither
of which has a syntactic attribute of $Z$. Thus, the expression "presented here" is non-negative, whereas "not presented here" is negative.

c) A "parenthetical expression" consists of the table entries delimited by paired table entries with punctuation attributes of left parenthesis and right parenthesis. A "parenthetical expression" also consists of the table entries between a serial comma or the beginning of the sentence and a parenthetical comma as determined by the subroutine CHCOMA. CHCOMA also identifies the first clause, if there is more than one.

d) If a sentence requires an antecedent, the preceding sentence is restituted if it had been previously deleted; this procedure is repeated until a sentence which does not require an antecedent is found. If more than three sentences need to be restituted, the present sentence is deleted and no restitution takes place. This procedure is never applied to the first sentence of the article.

e) A verb is considered to be in a sentence if the syntactic attributes $V$, $W$, or $X$ can be found in the sentence.

RULE 5: The semantic attribute $A$ identifies items which are undesirable in an abstract; if an $A$ occurs in a sentence the sentence will be removed, and previous sentences may also be removed if they are "weak" sentences.

\[
\text{if (non-negative } A \text{ in sentence)} \quad \text{then }^1 \\
\quad \cdot \text{ remove parenthetical expressions;} \\
\text{if (C in first clause)} \quad \text{then }^2
\]
if (previous sentence gone) then
/* undesirable sentence that requires an antecedent and
the previous sentence is gone */
delete sentence;
else
/* undesirable sentence that requires an antecedent and
the previous sentence stayed */
if (previous sentence was weak) then
delete this sentence and
previous sentence;
else
delete this sentence;
else
/* undesirable sentence that does not require an antecedent */
delete sentence;
else;

Notes:

a) A sentence is considered "weak" in this rule if it does not
have a semantic attribute of $I$ or $K$.

RULE 6: Sentences containing question marks and equal signs are
deleted. This prevents equations and questions from appearing in the
abstract.

RULE 7: The semantic attribute $K$ indicates importance, but not as
strongly as the attribute $I$.

if (non-negative $K$ in sentence) then
if (3 or more B's in sentence) then
delete sentence; else;
/* moderately important sentence */
remove parenthetical expressions;
apply coherence criteria as in RULE 4;

RULE 8: If there is no I, A, K, question mark, or equal sign in the
sentence, remove parenthetical expressions.

RULE 9: If there is a semantic attribute of B in the first clause,
the sentence is deleted. Since this rule applies after parenthetical
expressions have been logically deleted, only the parts of the first
clause that remain are examined.

RULE 10: The semantic attributes E and L indicate quantitative or
qualitative data; if these attributes occur early in the first clause,
they may indicate detailed information not suited for the abstract.

    if (E or L occur as the first or second word
        of the first clause) then

        /* the coherence criteria requiring a strong antecedent
           comprise the next set of if statements */

        if (C in first clause) then
            if (previous sentence gone) then

                /* detailed information that requires an antecedent
                   and the previous sentence is gone */
                delete sentence;

            else then

                /* detailed information that requires an antecedent
                   and the previous sentence stayed */

                if (previous sentence was strong) then
if (verb in sentence) then
sentence stays;
else
delete sentence;
else
/* detailed information that does not require an antecedent */
delete sentence;
else 2
else 4
else
/* detailed information that does not require an antecedent */
delete sentence;
else 1;

Notes:
a) A sentence is "strong" if it has a table entry with a semantic attribute of 1.

RULE 11: The semantic attribute G indicates relevance to the title. If there is a G in the sentence, coherence criteria are applied as in RULE 4.

RULE 12: Quantitative information, i.e., the semantic attribute E, occurring after the first two words in the clause is subject to the coherence criteria requiring a strong antecedent as used in RULE 10. Notice that if the attribute E occurs in the first two words of the first clause (RULE 10) it overrides relevance to the title, whereas if it occurs later in the first clause, relevance to the title has precedence.

RULE 13: The present sentence is matched against the previous sentence by using the RELEVANT subroutine. RELEVANT introduces the
semantic attribute $G$ into table entries of words which have no attributes and which co-occur in both sentences. It should be noted that since relevance to the title has been handled in RULE 11, any $G$'s encountered in this rule arise solely from words that co-occur in the present and the previous sentences.

$$\text{if} \ (G \ \text{in sentence})$$

$$\text{if} \ (\text{previous sentence gone}) \ \text{then}$$

$$\text{delete sentence}; \quad \text{else}$$

$$\text{else};$$

RULE 13 applies when there are no semantic keywords which provide a sound basis for deleting the sentence or for allowing it to remain. Words co-occurring in adjacent sentences make it reasonable to assume that the previous sentence is necessary for coherence; thus, the sentence is deleted if the previous sentence has been deleted.

RULE 14: This is a coherence rule similar to RULE 13.

$$\text{if} \ (C \ \text{in first clause})$$

$$\text{if} \ (\text{previous sentence gone}) \ \text{then}$$

$$\text{delete sentence}; \quad \text{else}$$

$$\text{else};$$

RULE 15: Modifying phrases introduced by words which have the semantic attribute $H$ are generally removed.

$$\text{if} \ (H \ \text{in first clause})$$

$$\text{if} \ (H \ \text{is first word of sentence}) \ \text{then}$$

$$\text{delete sentence}; \quad \text{else}$$
The following rules apply to clauses other than the first; the rules refer to some of the semantic attributes used above but they affect the clauses rather than the whole sentence.

**RULE 16:**

```plaintext
if (no more clauses in sentence) then
  if (verb in sentence) then
    sentence stays;
  else
    delete sentence;
else
  get next clause;
```

**RULE 17:** If the semantic attributes E or L occur as the first or second word of the clause, the clause is deleted.

**RULE 18:** If the semantic attribute B occurs in the clause, the clause is deleted.

**RULE 19:** If the semantic attribute H occurs in the clause, deletion occurs from the location of the H to the end of the clause.
CONCLUSIONS

The abstracts obtained by application of the algorithms presented here are of sufficiently good quality to indicate that large-scale testing of the methods of the automatic abstracting system is warranted. Our results to date indicate that abstracts can be produced automatically at costs comparable to those manually produced, although economic feasibility can only be ascertained through large-scale testing.

On a test scale we have obtained 80 to 90 percent reduction of text without using the length of the abstract as a criterion in the abstracting algorithm and without changing the dictionary. The small size of the dictionary (700 entries) is due, in a large part, to the fact that it is possible to reject sentences by referring to a small list of frequently occurring words, whereas selection of sentences requires a long list of "desirable" words.

The IBM 360 programs require approximately 15,000 bytes of storage. The data structure has been assigned a storage capacity sufficient to process articles of over 5,000 words; 40,000 bytes are used to store the text and 60,000 bytes are used for the table. The SEMANTIC program is designed for ease of modification, rather than for speed; the cost of abstracting, however, is less than half a cent per word input.

It is expected that in a large-scale system the data structure will constitute a by-product which can be used in text searching and in the production of indexes. The input costs could be significantly reduced by using machine-readable text available from computerized
typesetting operations. Investigations along these lines are currently being carried out.
BIBLIOGRAPHY


APPENDIX

Only the listings of the programs CHCOMA, MAIN, SEMANTIC, and WORDCTRL are presented here; listings of the other programs have been presented by Salvador (9).

Sample Computer Produced Abstract p. 59
Dictionary p. 60
Program Listings
   CHCOMA p. 73
   MAIN p. 76
   SEMANTIC p. 82
   WORDCTRL p. 94
MOSQUITOES FEEDING ON INSECT LARVAE. #P. HARRIS, D. P. RIORDAN, D. COOKE. SCIENCE VOL. 164, NO. 3876, APR 11, 1969# OUR RESULTS SHOW THAT IN THE LABORATORY SOME MOSQUITOES FEED ON INSECT LARVAE AND PRODUCE VIABLE EGGS AS A RESULT; THEY ARE ATTRACTED TO THE LARVAE AND EVIDENTLY RECOGNIZE THEM AS HOSTS. LABORATORY-PROPAGATED Aedes aegypti AND Culex tarsalis WERE KEPT IN CAGES, 22 BY 22 BY 37 CM, AT 26 DEGREES C, 65% RELATIVE HUMIDITY, AND A 14-HOUR PHOTOPERIOD. THE LIVING INVERTEBRATES WERE PLACED IN THE CAGE WITH THE MOSQUITOES, AND THE CODDLED INSECTS WERE TIED TO THE CAGE WALLS. WHEN CODDLED Celerio euphorbiae WERE EXPOSED FOUR AT A TIME FOR 1 HOUR, SIX, THREE, ONE, AND ZERO Aedes aegypti FED WHEN THE AVERAGE LARVAL WEIGHTS WERE 0.69, 0.16, 0.04, AND 0.03 G OR LESS, RESPECTIVELY. IN SEROLOGICAL SURVEYS THE CUT CONTENTS OF WILD-CAUGHT MOSQUITOES ARE TESTED ONLY IF DARK OR RED, AND THEN ONLY AGAINST VERTEBRATE ANTISERUMS. IN OPEN WINDSWEPT REGIONS, MOSQUITOES SHELTER IN CLUMPS OF DENSE VEGETATION WHERE THEY ARE CLOSE TO FEEDING LARVAE; THUS THE MOSQUITOES DO NOT NECESSARILY HAVE TO COVER DISTANCES GREATER THAN THOSE IN OUR EXPERIMENTS TO FIND LARVAE.
A NUMBER OF
A POINT OF
ABLE
ABOUT
ABOVE
ACCORDING
ACCORDINGLY
ACKNOWLEDGE
ACTUAL
ACTUALLY
ADDITIONAL
ADVANTAGES
AFTER
AGAIN
AGO
AGREEMENT
ALL
ALLOWING
ALSO
ALTERNATELY
ALTERNATIVE
ALTERNATIVELY
ALTERNATIVES
ALTHOUGH
ALWAYS
AN
AND
ANNUAL CONVENTION
ANNUAL REVIEW
ANOTHER
ANY
APPEAR
APPEARS
APPRECIATION
APPROACHES
ARE
ARE A NUMBER
ARE BEING
ARE
AREA
AS A RESULT
AS
AS WELL AS
AS
ASPECT
ASSIGNED
ASSIGNMENT
ASSUME
ASSUMED
ASSUMES
ASSUMING
ASSUMPTION
AT LEAST
CHCOMA   CSECT
STM 14,12,12(13)
BALR R10,0
USING *,R10
ST R13,SAVEAREA+4
LA R2,SAVEAREA
ST R2,8(R13)
LR R13,R2
B SAVEAREA+72
SAVEAREA DS 18F
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
LM R7,R9,0(R1)
L R7,0(R7)
L R8,0(R8)
LA R6,9
MVI PREV,X'00'
COMMA CLI 5(R7),C','
BE FOUND
NEXT LA R7,8(R7)
CR R7,R8
BNH COMMA
OUT MVI 0(R9),X'00'
L R13,SAVEAREA+4
RETURN (14,12)
FOUND ST R7,0(R9)
BCT R6,CHECK
B OUT
SER MVI 0(R9),X'02'
B INCR
NORM MVI 0(R9),X'01'
INCR MVC PREV,0(R9)
LA R9,4(R9)
B NEXT
CHECK CLI 13(R7),C'C'
BNE XX
CLI PREV,X'02'
BNE NORM
MVI 0(R9),X'02'
B INCR
XX LR R2, R7
XX1 S R2, =F'8'
CLI 4(R2), C'J'
BE XX1
CLI 4(R2), C'B'
BNE PRNN
MVI 0(R9), X'03'
C R6, =F'8'
BE INCR
B PARN1
PRNN CLI 13(R7), C'N'
BE NORM
CLI 13(R7), C'O'
BE PARN
CLI 13(R7), C'V'
BE PARN
CLI 13(R7), C'W'
BE PARN
CLI 13(R7), C'X'
BNE CALL
PARN C R6, =F'8'
BE SER
MVI 0(R9), X'03'
PARN1 CLI PREV, X'02'
BE INCR
LR R2, R9
S R2, =F'4'
MVI 0(R2), X'02'
B INCR
CALL LA R2, 61(R7)
LA R3, 5(R8)
CR R2, R3
BL CA1
ST R3, CHEND
B CA2
CA1 ST R2, CHEND
CA2 MVC ROLES(2), =C'C,' LA R1, 13(R7)
ST R1, CHBEG
MVI AJA, X'00'
CALL CHECKROL, (CHBEG, CHEND, ROLES)
CLI ROLES, X'00'
BE CA3
OI AJA, X'04'
MVC LAST(4), CHBEG
CA3 LA R1, 13(R7)
ST R1, CHBEG
MVC ROLES(6), =C'WXYZ'
CALL CHECKROL, (CHBEG, CHEND, ROLES)
CLI ROLES, X'00'
BE CA4
OI AJA, X'08'
CA4
SR  R1, R1
IC  R1, AJA
B   CA5 (R1)

CA5
B   NORM
B   SEP
B   NORM
CLC  CHBEG, LAST
BL  NORM
B   SER
DC  C 'CONSTANTS'

AJA
DS  XL1
PREV  DS  CL1
LAST  DS  F
CHBEG  DS  F
CHEND  DS  F
ROLES  DC  3F'0'
END
MAIN CSECT
ENTRY PRINTER
STM 14,12,12(13)
BALR R10,0
USING #,R10
ST R13,SAVEAREA+4
LA R2,SAVEAREA
ST R2,8(R13)
LR R13,R2
B SAVEAREA+72
SAVEAREA DS 18F
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
OPEN (CARD, (INPUT), PRINTER, (OUTPUT))
TOSIZE EQU 4000C
TBSIZE EQU 60000
START L R5,=A(IOAREA)
NADA GET CARD
CLC 0(2,R1),=C'$$'
BE LOOP10
MVC 0(80,R5),0(R1)
LA R5,80(R5)
C R5,=A(IOAREA+IOSIZE)
BL NADA
FL1 PUT PRINTER,CHAN1
MVC LINE,BLANKS
MVC LINE(42),=C'IOAREA
PUT PRINTER,CC
FLUSH GET CARD
CLC 0(2,R1),=C'$$'
BE START
B FLUSH
LOOP10 MVC OPTIONS(78),2(R1)
L02 BCTR R5,0
CLI 0(R5),C'.'
BNE L02
ST R5,ENDPERIOD
L R3,=A(IOAREA+IOSIZE)
MVI 0(R3),X'00'
CALL MOVE,(IOAREA+IOSIZE+1,IOAREA+IOSIZE,LEN)
L R5,=A(IOAREA)
L R6,=A(IOAREA+IOSIZE)
SR R1,R1
L R11,=F'0'
LOOP11 C R6,=A(IOAREA+IOSIZE+TBSIZE)
BNL FL1
TRT 0(256,R5),TRTBLANK
BC 6,LOOP12
LA R5,256(R5)
B LOOP11

LOOP12 LR R5,R1
LOOP13 TRT 0(256,R5),TRTBL
BC 6,LOOP14
LA R5,256(R5)
B LOOP13

LOOP14 CLI 0(R1),C'
BNE CHAR1
B STW

CHAR1 TRT 0(1,R1),CHAR
BC 6,PHONY
CR R5,R1
BNE STWDCH
B SEQCHAR

PHONY CLI 0(R1),C'
BE LEFTBLK
CLI 0(R1),C'<
BE LEFTBLK
CLI 0(R1),C'
BNE RIGHTBLK

HYPHEN LR R3,R1
BCTR R3,0
CLI 0(R3),C'
BE RIGHT
CLI 0(R3),C'A'
BL FORGET
CLI 0(R3),C'Z'
BH FORGET

RIGHT CLI 2(R3),C'
BE CR1
CLI 2(R3),C'A'
BL FORGET
CLI 2(R3),C'Z'
BH FORGET
B CR1

RIGHTBLK C R1,ENDOR
BE CR1
STM R1,R2,DLB
TRT 1(1,R1),TRTBL
LM R1,R2,DLB
BC 8,FORGET

CR1 CR R5,R1
BE SEQCHAR
B  STWDCH

LEFTBLK LR  R3,R1
     BCTR R3,0
     CLI 0(R3),C
     BE  SEQCHAR

FORGET LA  R1,1(R1)
     TRT 0(256,P1),TRTBL
     BC 6,LOOP14
     LA  R1,256(R1)
     B  FORGET

STWDCH ST  R1,8(R6)
     MVI 8(R6),X'01'
     STC R2,13(R6)
     STC R2,12(R6)
     STH R11,6(R6)
     LA  R11,1(R11)
     STH R11,14(R6)
     LR  R3,R1
     SR  R3,R5
     ST  R5,0(R6)
     STC R3,0(R6)
     C  R1,ENDPEROD
     BNL  SORT
     LA  R1,1(R1)
     LR  R5,R1
     LA  R6,16(R6)
     LA  R11,1(R11)
     B  LOOP11

SEQCHAR STH R11,6(R6)
     ST  R5,0(R6)
     MVI 0(R6),X'01'
     STC R2,5(R6)
     STC R2,4(R6)
     C  R5,ENDPEROD
     BNL  SORT
     LA  R5,1(R5)
     LA  R6,8(R6)
     LA  R11,1(R11)
     B  LOOP11

STWD LR  R3,R1
     SR  R3,R5
     ST  R5,0(R6)
     STC R3,0(R6)
     STH R11,6(R6)
     LA  R1,1(R1)
     LR  R5,R1
     LA  R6,8(R6)
     LA  R11,1(R11)
     B  LOOP11

PRINT NOGEN

SORT ST  R11,ULTIMA
     CALL SORT,(IOAREA+IOSIZE,ULTIMA)
     CLI OPTIONS,C'**
RETURN (14, 12)

TRTTBL
DC 256X'00'
ORG TRTTBL+C'
DC C'
ORG TRTTBL+C'
DC 7AL1(*-TRTTBL)
ORG TRTTBL+C'
DC 8AL1(*-TRTTBL)
ORG TRTTBL+C'
DC 5AL1(*-TRTTBL)
ORG TRTTBL+C'
DC 6AL1(*-TRTTBL)
ORG

TRTBLANK
DC 256AL1(*-TRTBLANK)
ORG TRTBLANK
DC X'FF'
ORG TRTBLANK+C'
DC X'00'
ORG

CHAR
DC 256X'00'
ORG CHAR+C'
DC 3AL1(*-CHAR)
ORG CHAR+C'
DC C'
ORG CHAR+C'
DC C-
ORG CHAR+C'
DC C'
ORG CHAR+C'
DC C'>
ORG

PRINTER
DCB DDNAME=PRINTER,DSORG=PS,MACRF=(PM)
CHAN1 DC X'88'
CC DC X'09'
LINE DS CL132
BLANKS DC CL132
CARD DCB DDNAME=CAPD,DSORG=PS,EODAD=EOJ,MACRF=(GL)
DC C'

CONSTANTS
FIRST DS XL1
IND DS XL1
LEN DC F'59999'
OPTIONS DS CL78
DBL DS D
ULTIMA DS F
ENDPERIOD DS F
DATA DS 2F
FORMAT DC C'JXOGXLDDE31'
DC C'TR1LA19L0OXL'
DDCARD DC C'1LS31XAI'
DC C'QLVVL30X'
LTORG DS OD
IOAREA DS 40000CL1
MACRO
&LABEL DELT &A,&B
&LABEL L R1,&A
LA R1,0(R1)
ST R1,DELBGN
L R1,&B
LA R1,0(R1)
ST R1,DELEND
BAL R11,DELETE
MEND
MACRO
&LABEL ROLCH &A,&B
&LABEL L R1,&A
LA R1,4(R1)
ST R1,CHBEG
L R1,&B
LA R1,4(R1)
ST R1,CHEND
CALL CHECKROL,(CHBEG,CHEND,ROLES)
MEND
MACRO
&LABEL ROLSY &A,&B
&LABEL L R1,&A
LA R1,5(R1)
ST R1,CHBEG
L R1,&B
LA R1,5(R1)
ST R1,CHEND
CALL CHECKROL,(CHBEG,CHEND,ROLES)
MEND
SEMA TITLE ' SEMANTIC'
SEMAN CSECT
STM 14,12,12(13)
BALR R10,0
USING *,R10
ST R13,SAVEAREA+4
LA R2,SAVEAREA
ST R2,8(R13)
LR R13,R2
B SAVEAREA+72
SAVEAREA DS 18F
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12  EQU  12
R13  EQU  13
R14  EQU  14
R15  EQU  15
PRINT NOGEN
LM  R7,R8,0(R1)
L  R8,0(R8)
STM  R7,R8,TABLE
SR  R6,R6
SE  MVC  6(Z,R7),-"L2'0"
C  R6,ULTIMA
BNL  SE1
LA  R7,8(R7)
LA  R6,1(R6)
B  SE
SE1  ST  R7,TABPEROD
MVI  END,"X'00"
ZAP  JULIO,"P'0"
L  R1,TABLE
S  R1,"F'8"
ST  R1,ESENTC
BAL  R11,PERIOD
SE2  L  R1,ESENTC
CLI  "12(R1),C'##"
BE  SE3
CLI  ROLES,C'##"'
BE  NOPFF
BAL  R11,PERIOD
B  SE2
NOREF  MVC  POUND2,"F'0"
MVC  ETITLE,ESENTC
B  SE4
SE3  MVC  ETITLE,ESENTC
LA  R1,20(R1)
ST  R1,CHBEG
LA  R1,1600(R1)
C  R1,TABPEROD
BH  SE31
ST  R1,CHEND
SE31  MVI  ROLES,C'##"'
CALL  CHECKROL,(CHBEG,CHEND,ROLES)
CLI  ROLES,"X'00"
BE  NOREF
L  R1,CHBEG
S  R1,"F'4"
ST  R1,POUND2
ST  R1,ESENTC
SE4  MVC  A,TABLE
MVC  B,ETITLE
L  R1,ESENTC
LA  R1,8(R1)
ST  R1,C
MVC  D,TABPEROD
BAL R11,RELEVANT
MVI YYYZ,X'00'
MVI YYONLY,X'00'
MVI BSE1,X'00'
MVI BSE2,X'00'
MVI BSE3,X'00'
MVI BSENTC,X'00'
CHECK TM END,X'01'
BO FLAREM
MVI X,X'00'
MVI EL,X'00'
MVI PAGRE,X'00'
ZAP K1,=P'0'
AP JULIO,=P'1'
MVC BSE3,BSE2
MVC BSE2,BSE1
MVC BSE1,BSENTC
MVC PRYYONLY,YYONLY
MVI YYONLY,X'00'
MVC PRYYYZ,YYYZ
MVI YYYZ,X'00'
BAL R11,PERIOD
MVI ROLES,C'I'
ROLCH BSENTC,ESENTC
CHE1 MVC YYYY,ROLES
MVC YYONLY,ROLES
MVC YYYY,ROLES
CLI ROLES,C'I'
BNE CHE2
LA R15,X1Y1
BAL R14,NEGTEST
MVI ROLES,C'A'
BAL R12,SCAN
B CHE1
CHE2 MVI ROLES,C'A'
ROLCH BSENTC,ESENTC
CHE3 CLI ROLES,C'A'
BNE CH1
LA R15,CHE4
BAL R14,NEGTEST
MVI ROLES,C'A'
BAL R12,SCAN
B CHE3
CHE4 MVI X,X'01'
B X1Y1
DELE DELT BSENTC,ESENTC
L R1,BSENTC
MVI 7(R1),X'01'
MVI PRVSEN,X'01'
MVI YYYZ,X'00'
MVI YYONLY,X'00'
B CHECK
CH1 MVC ROLES(2),=C'?='
ROLCH BSENC, ESENTC
CLI ROLES, X'00'
BNE DELE
MVI ROLES, C'K'
ROLCH BSENC, ESENTC
MVC YYYES, ROLES

CHE5
CLI ROLES, C'K'
BNE X1Y1
LA R15, CHE6
BAL R14, NEGTEST
MVI ROLES, C'K'
BAL R12, SCAN
B CHE5

CHE6
MVC YYYZ, ROLES
X1Y11
ZAP IND1, =P'0'
MVC TEMPI, BSENTC
MVI ROLES, C'B'

CH2
ROLCH TEMPI, ESENTC
CLI ROLES, X'00'
BE X1Y1
AP IND1, =P'1'
CP IND1, =P'3'
BNE DELE
L R1, CHBEG
LA R1, 4(R1)
ST R1, TEMPI
B CH2

NEGTEST
L R1, CHBEG
SH R1, =H'7'
CLI O(R1), C'Z'
BCR 8, R14
SH R1, =H'8'
CLI O(R1), C'Z'
BCR 8, R14
BR R15

SCAN
L R1, CHBEG
LA R1, 4(R1)
ST R1, CHBEG
ROLCH CHBEG, ESENTC
BR R12

XY2
MVI PACORE, X'01'
XY22
CALL CHCOMA, (BSENTC, ESENTC, COMAS)
MVC BPHRAS, BSENTC
LA R1, COMAS
LA R3, PROCED1
LA R4, GOOD1

LP1
CLI O(R1), X'00'
BCR 8, R3
CLI O(R1), X'01'
BCR 8, R4
CLI O(R1), X'03'
RE Z3

LP2
LA R1, 4(R1)
Z3
B LP1
C R1,=A(COMAS)
SNE Z4
MVC TEMP1,BSENTC
MVC TEMP2,0(R1)
B Z5
Z4
MVC TEMP2,0(R1)
LR R2,R1
S R2,=F'4'
MVC TEMP1,0(R2)
Z5
LR R5,R1
MVI KILL,X'01'
DELT TEMP1,TEMP2
LR R1,R5
B LP2
PROCE01
MVI EODEN,X'01'
MVC EPHRAS,ESENTC
B Z6
GOOD1
MVI EODEN,X'00'
MVC EPHRAS,0(R1)
ST R1,TABCOM
Z6
CLI YYYY,X'00'
BNE XY44
CLI X,X'00'
BNE XY44
MVI ROLES,C'B'
ROLCH BPHRAS,EPHRAS
CLI ROLES,C'B'
BE DELE
L R1,BSENTC
CLI 4(R1),C'E'
BE Z7
CLI 4(R1),C'L'
BE Z7
CLI 12(R1),C'E'
BE Z7
CLI 12(R1),C'L'
BE Z7
B G
Z7
MVI EL,X'01'
B XY44
G
MVI ROLES,C'G'
ROLCH BSENTC,ESENTC
MVC YYYY,ROLES
CLI ROLES,C'G'
BE XY44
X1Y1
MVC TEMP2,BSENTC
XY1
MVI ROLES,C''
ROLCH TEMP2,ESENTC
CLI ROLES,C''
BNE FONY
L R3,CHBEG
S R3,=F'4'
ST    R3, TEMP1
LA    R3, 8(R3)
AP    K1, = P'1'
SIGUE ST    R3, BPAREN
MVC    ROLES(2), = C'()
ROLCH BPAREN, ESENTC
CLI    ROLES, X'00'
BE    FONY
CLI    ROLES, C')'
BE    SIG1
AP    K1, = P'1'
L    R3, CHBEG
LA    R3, 4(R3)
B    SIGUE
SIG1 SP    K1, = P'1'
CP    K1, = P'0'
BE    REMPAR
L    R3, CHBEG
LA    R3, 4(R3)
B    SIGUE
REMPAR L    R1, CHBEG
S    R1, = F'4'
ST    R1, EPAREN
MVC    BPAREN, TEMP1
MVI    KILL, X'01'
DELT    BPAREN, EPAREN
REMP L    R3, EPAREN
LA    R3, 8(R3)
C    R3, ESENTC
BNL    FONY
ST    R3, TEMP2
B    XY1
COREPA MVC    TEMP3, BPAREN
MVI    ROLES, C', '
ROLCH TEMP3, EPAREN
CLI    ROLES, C', '
BNE    REMP
L    R1, CHBEG
MVC    0(2, R1), = X'0101'
S    R1, = F'4'
ST    R1, TEMP3
B    COR
FONY TM    PACORE, X'01'
BZ    XY2
MVI    ROLES, C'E'
ROLCH BSENTC, EPHRAS
CLI    ROLES, C'E'
BNE    XY4
MVI    EL, X'01'
B    XY44
XY4 CP    JULIO, = P'1'
BE    IFYY1
MVC    A, BSENTC
MVC B, ESENTC
MVC C, RSE1
L R1, BSENTC
S R1, =F'8'
ST R1, D
BAL R11, RELEVANT
MVI ROLES, C'G'
ROLCH C, D
CLI ROLES, C'G'
RNE XY44
TM PVSEN, X'01'
RO DELE

XY44 CP JULIO, =P'1'
BE IFYY1
MVI ROLES, C'C'
ROLCH BSENTC, EPHRAS
CLI ROLES, C'C'
RNE IFYY1
TM PVSEN, X'01'
BZ IFYY1
EUSKAL CLI YYYES, X'00'
BE DELE
LA R4, 3
LA R12, RSE1
LA R5, BSENTC
EUSKAL1 MVC RSTR1, 0(R12)
L R1, 0(R5)
S R1, =F'8'
ST R1, RSTR2
BAL R11, RESTORE
CALL CHCOMA, (RSTR1, RSTR2, COMAS2)
LA R1, COMAS2

LLPP11 CLI 0(R1), X'00'
BE NEXTXY
CLI 0(R1), X'01'
BE YESCOMA
LA R1, 4(R1)
B LLPP11

YESCOMA MVC RSTR2, 0(R1)
MVI ROLES, C'C'
ROLCH RSTR1, PSTR2
CLI ROLES, C'C'
BE EUSKAL2
B XY9

EUSKAL2 LA R12, 4(R12)
LA R5, 4(R5)
BCT R4, EUSKAL1
DELT BSE3, ESENTC
MVI PVSEN, X'01'
B CHECK

RESTORE L R1, RSTR1
RES1 MVI 6(R1), X'F9'
C R1, RSTR2.
BE Z666
CLI 12(R1),CL'L'
BNE Z77
Z666 L R1,BPHRAS
BAL R11,FIXPC
Z77 MVI ROLES,C'B'
ROLCH BPHRAS,EPHRAS
CLI ROLES,C'B'
BNE Z88
Z88 L R1,BPHRAS
BAL R11,FIXPC
MVI ROLES,C'H'
ROLCH BPHRAS,EPHRAS
CLI ROLES,C'H'
BNE NEXCOM
L R1,CHBEG
S R1='F'4'
BAL R11,FIXPC
B NEXCOM
XY9 MVC TEMP,BSENTC
XY99 MVC ROLES(3),='CVWX'
ROLSY TEMP,ESENTC
CLI ROLES,X'00'
BE DELE
L R1,CHBEG
CLI 1(R1),C'D'
BNE XY100
LA R1,3(R1)
ST R1,TEMP
B XY99
XY100 MVI PRVSEN,X'00'
B CHECK
FLAREM L R13,SAVEAREA+4
RETURN (14,12)
PERIOD STM R0,R15,SAVE
L R1,ESENTC
LA R1,8(R1)
ST R1,BSENTC
LA R1,4(R1)
ST R1,CHBEG
L R1,TABPEROD
LA R1,4(R1)
ST R1,CHEND
MVC ROLES(2),='C'4;''
CALL CHECKROL,(CHBEG,CHEND,ROLES)
L R1,CHBEG
PEO CLI 8(R1),C'.'
BNE PE1
LA R1,8(R1)
C R1,CHEND
BE PE3
B PE0
PE1 S R1,='F'4'
ST R1, ESENTC
CLC CHBEG, CHEND
BNE PE5
PE4 MVI END, X'01'
B PE5
PE3 S R1, =F'4'
ST R1, ESENTC
B PE4
PE5 LM RO, R15, SAVE
BR R11
DELETE ST R1, SAVE
L R1, DE LBGN
DELOOP C R1, DELE ND
BH DEL1
CLI K ILL, X'00'
BE DELO
MVC 4(2, R1), =C'DD'
DELO MVI 6(R1), C'D'
LA R1, 8(R1)
B DE LOOP
DEL1 L R1, SAVE
MVI K ILL, X'00'
BR R11
RELEVANT STM RO, R15, SAVE
L R6, A
ARROW1 C R6, B
BNL OUTTT
CLI 5(R6), C'S'
BE ARROW11
CLC 4(2, R6), =XL2'0'
BNE AR1
ARROW11 L R8, C
AR2 C R8, D
BNL AR1
CLC 0(1, R8), 0(R6)
BNE AR3
L R11, 0(R6)
L R12, 0(R8)
SR R1, P1
IC R1, 0(R6)
BCTR R1, 0
STC R1, AR4+1
AR4 CLC 0(0, R11), 0(R12)
BNE AR3
CLC A, TABLE
BNE AR5
MVI 5(R6), C'S'
MVI 5(R8), C'S'
AR5 MVI 4(R6), C'G'
MVI 4(R8), C'G'
AR3 LA R8, 8(R8)
B AR2
AR1 LA R6, 8(R6)
B  ARROW1
     OUTTT LM RO,R15,SAVE
     BR R11
     FIXPC STM RO,R15,SAVE
     C R1,BPHRAS
     BNE FIX1
     MVI SKIP,X'01'
     S R1,=F'8'
     ST R1,TEMP1
     B FIX2
     FIX1 ST R1,TEMP1
     FIX2 ST R1,TEMP1
     S R1,EPHRAS
     ST R1,TEMP2
     DELT TEMP1,TEMP2
     TM SKIP,X'01'
     LM RO,R15,SAVE
     BR NEXCOM
     ARROW1 RO,R15,SAVE
     NEXCOM

TABLE DS F
ULTIMA DS F
TABPERIOD DS F
CHBEG DS F
CHEND DS F
ROLES DC 3F'0'
POUND2 DS F
ETITLE DS F
SAVE DS 16F
A DS F
B DS F
C DS F
D DS F
END DS XL1
YYYES DS XL1
PRVSEN DS XL1
PACORE DS XL1
ENSENTC DS F
RSENTC DS F
BSE1 DS F
BSE2 DS F
BSE3 DS F
DELRGN DS F
DELEND DS F
COMAS DS 10F
COMAS2 DS 10F
 TEMP DA DS F
 TEMP1 DS F
 TEMP2 DS F
 TEMP3 DS F
BPHRAS DS F
EPHRAS DS F
EOSEN DS XL1
IND1  DS  XL1
K1    DS  PL2
BPAREN DS  F
EPAREN DS  F
RSTR1 DS  F
RSTR2 DS  F
TABCOM DS  F
JULIO DS  PL6
SKIP  DS  XL1
YYYZ  DS  XL1
PRYYYZ DS  XL1
PRYYONLY DS  XL1
YYONLY DS  XL1
KILL  DC  X'00'
EL    DS  XL1
X     DS  XL1
END
WORDCTRL CSECT
STM  14,12,12(13)
BALR R10,0
USING *,R10
ST  R13,SAVEAREA+4
LA  R2,SAVEAREA
ST  R2,8(R13)
LR  R13,R2
B   SAVEAREA+72
SAVEAREA DS  18F
R0  EQU  0
R1  EQU  1
R2  EQU  2
R3  EQU  3
R4  EQU  4
R5  EQU  5
R6  EQU  6
R7  EQU  7
R8  EQU  8
R9  EQU  9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
LM  R7,R8,0(R1)
L   R8,0(R8)
ST  R8,ULTIMA
TBL EQU  7
K1  EQU  9
MVI K0,X'00'
L   K1,=F'-1'
OPEN (STPLIST,(INPUT))
READ2 ZAP RIC,=P'0'
ZAP IAI,=P'1'
GET STPLIST
LA  R4,TABLILLA
LR  R3,R1
TRT TPT O(60,R3),BLKSTR
LR  R2,R1
SP  R1,R3
ST  R3,0(R4)
STC R1,0(R4)
CLI O(R2),C'*'
BE  LOOP2
LA  R4,4(R4)
LA  R2,1(R2)
LR  R3,R2
AP  IAI,=P'1'
B   TRT
LOOP2 MVC WORCOD(1),1(R2)
MVC WORCOD+1(1),3(R2)
LOOP16
ZAP LEON,=P'1'
LA K1,1(K1)
C K1,ULTIMA
BH MALT2
ST K1,K11
LR R6,K1
SLL R6,3
LH R8,6(R6,TBL)
SLL R8,3
LA R5,0(TBL,R8)
CLI 5(R5),X'00'
BE REPITE
CLI 5(R5),X'80'
BL LOOP16
REPITE
SK R1,R1
CLC TABLLILA(1),0(R5)
IC R1,0(R5)
BH LOOK
TC R1,TABLLILA
LOOK
BCTR R1,0
L R11,0(R5)
L R12,TABLLILA
BE MATCH
BH REST
BL LNLLOW
COMP CLC 0(0,R11),0(R12)
MATCH EX R1,COMP
BE PASA
BH MAT2
MAT1 CP RIC,=P'0'
BE LOOP16
B FOLLON
MAT2 CP RIC,=P'0'
BE LEESTO
S FOLLON
REST EX R1,COMP
BH MAT2
B MAT1
LNLLOW EX R1,COMP
BL MAT1
B MAT2
PASA AP RIC,=P'1'
CP RIC,=P'1'
BNE PASA1
ST K1,KK1
MVC CTR,=F'0'
PASA1 CP IA1,=P'1'
BE EMPTY
LA R12,TABLLILA
LR R11,R5
CLI 4(R5),C'J'
BE PASA8
CLI 4(R5),X'00'
BNE PASA9
PASA8 MVI 4(R5),X'FF'
LA RO,1
A RO,CTR
ST RO,CTR
PASA9 CLI WORCOD+1,X'00'
BE CLEARKO
CLI 5(R5),X'00'
BNE CLEARKO
PASA10 MVC 5(1,R5),WORCOD+1
CLEARKO MVI KO,X'00'
B LOOP16
LEESTO MVI KO,X'01'
B READ2
FOLLOW ST K1,K12
SR R2,R2
TRT WORCOD(1),ABIK
BC 8,ASIS
L RO,CTR
LA R15,500
C R15,ULTIMA
BL USECOUNT
LA R15,1000
MR R14,RO
D R14,ULTIMA
LR RO,R15
USECOUNT B USECOUNT(R2)
B A
B B
B I
B K
A CH RO,=H'7'
BNH ASIS
MVI WORCOD,C'B'
B ASIS
B CH RO,=H'7'
BNH ASIS
MVI WORCOD,C'E'
B ASIS
I CH RO,=H'4'
BNH ASIS
MVI WORCOD,C'K'
CH RO,=H'8'
BNH ASIS
MVI WORCOD,C'G'
B ASIS
K CH RO,=H'4'
BNH ASIS
MVI WORCOD,C'G'
CH RO,=H'8'
BHN ASIS
MVI WORCOD,X'00'
ASIS L K1,KK1
BCTR K1,0
ASISLOOP LA K1,1(K1)
C K1,K12
BNL FOLL02
LR R6,K1
SLL R6,3
LH R8,6(R6,TBL)
SLL R8,3
LA R5,0(TBL,R8)
CLI 4(R5),X'FF'
BNE ASISLOOP
MVC 4(1,R5),WORCOD
B ASISLOOP
FOLL02 MVI KO,X'00'
L K1,KK1
BCTR K1,0
B READ2
HALT2 CP RIC,=P'0'
BNE FOLLON
EOJ CLOSE (STPLIST)
L R13,SAVEAREA+1
RETURN (14,12)
DC C'CONSTANTS'
KK1 DS F
WORCOD DS CL2
LEON DS PL2
IAI DS PL2
RIC DS PL2
BRUJA DS PL2
BRUJO DS PL1
KO DS XL1
ULTIMA DS F
K11 DS F
TABLLILLA DS 15F
TRTBL DC 256AL1(*-TRTBL)
ORG TRTBL+C'
DC X'00'
ORG
BLKSTR DC 256X'00'
ORG BLKSTR+C'
DC C'*
ORG BLKSTR+C'*
DC C'*
ORG
ABIK DC 256X'00'
ORG ABIK+C'A'
DC X'0408'
ORG ABIK+C'I'
DC X'0C'
ORG ABIK+C'K'
<table>
<thead>
<tr>
<th>DC</th>
<th>ORG</th>
<th>K12</th>
<th>DS</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>K12</td>
<td>DS</td>
<td>CTR</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>STPLIST</td>
<td>DCR</td>
<td>STPLIST</td>
<td>MACRF=(GL), DSORG=PS, EODAD=EOJ</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summarize in fifty words or less the purpose and principal conclusions of your thesis

This thesis discusses the design of an automatic abstracting system. Methods for selecting material to create an abstract and a data structure which permits effective manipulation of the data are studied. Abstracts are produced solely by applying coherence and contextual inference criteria in the selection and rejection of sentences from the original document. Abstracts which are 10 to 20 percent the size of the original documents have been produced with the aid of a dictionary of approximately 700 entries.