teristic of the particular output being processed and must be established for the advantage of the user. In any event, the layout of each item in the page being produced is greatly aided by passing to the composition program whatever description of the record is necessary for proper layout. Such information is best obtained and added to the input record by the input processing. For example, to lay out the item in the space remaining on a page, the number of lines of each information type which is subject to variation is required, such as lines involved in the title, number of author lines, reference lines, abstract lines, etc. Not only can the total length be computed quickly at output time without testing the content of each line, but also discrete directions can be provided to the output program on how to proceed to output the information from each record area.

It generally is more efficient to program a line count of the space remaining on a page than to call system routines for each line printed to obtain the line number.

In many instances it is advantageous to provide, at input time, certain prefabricated lines produced from individual items which must be kept separate for sorting and record selection purposes. When upper/lower case records are involved, it is also necessary to provide both all upper case fields for sorting and upper/lower case fields for printing. The translation procedure for converting to all upper case is provided preferably at input time, and is another example of output processing that begins at input time.

On reviewing some of the requirements for good output processing, it is apparent that the purpose of good input is to provide fully processable records for output. Thus input ends where rearrangement of records for output begins. Input provides the bridge between the input document and the output products. The outputs provide the bridge between the computerized information system and its user.

LITERATURE CITED

(1) Skolnik, H., "The Multiterm Index: A New Concept in Information Storage and Retrieval," *J. Chem. Doc.*, **10**, 81–84 (1970).

(2) Skolnik, H., and B. E. Clouser, "Designing an Information Awareness and Retrieval System for Chemical Propulsion Literature," *J. Chem. Doc.*, **11**, 39–43 (1971).

(3) Skolnik, H., "The What and How of Computers for Chemical Information Systems," *J. Chem. Doc.*, **11**, 185–189 (1971).

(4) Skolnik, H., and W. L. Jenkins, "Evaluation of the IBM Administrative Terminal System and Magnetic Tape Selectric Typewriter for Text Processing," *J. Chem. Doc.*, **11**, 170–173 (1971).

Character Sets†

DONALD F. RULE

Chemical Abstracts Service, Columbus, Ohio 43210

Received November 13, 1974

Historically, character sets have grown and developed with the growth of the "frame" or byte size of data processing equipment. Starting with the 5 level (bit) Baudot codes, the popular industry codes have progressed through the 6-bit BCD, the 7-bit ASCII, the 8-bit EBCDIC. In keeping with the *convenient* code size limit of 2^n where n is the byte size of the computer, I/O hardware offerings and programming languages have generally constrained themselves to available byte-size environments. Fortran uses a character set of 49 symbols; Cobol uses a set of 51 symbols; PL/1 uses 60 symbols; and IBM 360/370 Assembler uses 51. Impact line printers have typically progressed through offerings of 48, 60, 120, and 240 characters. CRT's have progressed through a series of offerings similar to impact printers.

The 8-bit version of ASCII plus the proposed code extension techniques for ASCII represent the most significant industrywide thrust toward information interchange that is not so dominated by the byte-size environment of the current generation of computers. These extension techniques provide a mechanism for defining "pages" of character sets and for declaring pages to be active. In a 7-bit environment, one page of 128 characters can be active at a given moment. In an 8-bit environment, two 128 character pages may be active at a given moment.

The character-set needs of many information systems involving large data bases exceed the byte size of today's industry codes (excluding extended ASCII). Character usage studies of the published literature in the disciplines of chemistry, biology, and math show that the character set size needed for (English-based) information processing is in the neighborhood of 1000 characters and growing slowly. It is unlikely that the needed set size for other disciplines such as medicine, law, and physics is much larger although the 1000 characters for each discipline might be somewhat different. These character-set sizes do not include typographical variations such as type style, type face, or point size. Such variations are useful in printed matter, but are not (should not be) part of the intended information content of machine-readable information data bases.

Input of large character sets on devices with a limited number of keys and discrete device generated output codes need not be a major problem if the system designer keeps two points in mind. First, most special symbols occur very infrequently. Second, many characters are simply shape or placement variations of some basic symbol. For example, A, a, $_a$, ^{a}, and *A* are all variations of a single basic symbol. With these points in mind, the system designer can develop input conventions which are simple and yet efficient in terms of average number of keystrokes per character. Software routines translate the input code of a particular device into the internal character code which should be constrained only by the byte environment and not by a particular input or output device.

Output of large character sets is getting easier. Photocomposers meet the printing need for large character sets. Some non-impact printers and a few CRT devices provide

†Summary of paper presented in the NAS/NRC "Conference on Large Data Bases," sponsored by the NAS/NRC Committee on Chemical Information, National Academy of Sciences, May 22-23, 1974.

for output or display in the range of 1000 different character representations.

Many problems (challenges) remain. Information processing requires more than a character set as the following hierarchy suggests:

Information
  Knowledge
    Language
      Code
        Alphabet
          Character
            Symbol
              Sign

Meaning is a function of context. The above hierarchy suggests the context of a character. The data element/record/file/base formats and the precise meanings of the individual elements are an integral part of the code and language of information transfer. Without this context we do not know the meaning of our character sets.

"A " What is it? Without more context, I don't know. Do you? Is it one allograph of the first letter of the English (Latin) alphabet? Is it the 10th digit in hexadecimal notation? Is it a variable in a mathematical equation? Is it one allograph of the first letter in the Greek alphabet? Is it a student's grade? Is it the graphic symbol representation of the 8-bit code "11000001" in the code called EBCDIC?

Data Compression of Large Document Data Bases[†]

H. S. HEAPS

Department of Computer Science, Concordia University, Sir George Williams Campus, Montreal, Canada

Received November 8, 1974

Consideration is given to a document data base that is structured for information retrieval purposes by means of an inverted index and term dictionary. Vocabulary characteristics of various fields are described, and it is shown how the data base may be stored in a compressed form by use of restricted variable length codes that produce a compression not greatly in excess of the optimum that could be achieved through use of Huffman codes. The coding is word oriented. An alternative scheme of word fragment coding is described. It has the advantage that it allows the use of a small dictionary, but is less efficient with respect to compression of the data base.

INTRODUCTION

The subject of data compression includes many different aspects. The present paper is concerned with compression of textual data such as might appear in document titles, abstracts, author names, and so forth. It might include numerical data, but the methods to be described do not take advantage of relations that might exist between different sets of numerical data.

A data base for document retrieval may be envisaged as a set of records that are divided into fields, some of which may be searched for the presence of terms as specified in a question statement. Each searchable field is associated with a particular attribute such as "author names," "title," "abstract," "keywords," and so forth. There are many different ways in which such data bases may be structured for storage on computer accessible files.

The simplest method of structuring the data base is in a sequential manner as shown in Figure 1. An obvious disadvantage of sequential storage of a large data base is that any search for the records that contain particular terms involves search through the entire data base, and this is apt to be very time-consuming and therefore costly.

A different means of structuring a document data base is through use of an inverted file. A further alternative is through use of a hierarchically structured dictionary designed with reference to a hierarchical scheme of document classification. A detailed comparison of the two alternatives from the point of view of their effect on computing efficiency has been made by Ein-Dor.[1]

The form of inverted file shown in Figure 2 proves convenient for illustration of the concepts to be discussed in the present paper. It consists of a term dictionary in which title terms, author names, keywords, and so forth are arranged in alphabetic, or other, order together with pointers to sets of document numbers, or accession numbers, that indicate those documents that contain the particular term. A search for the documents that contain specified terms, or satisfy a particular question logic, involves application of logic operations to the appropriate sets of document numbers.

Once the pertinent document numbers have been determined, the document texts may be read from the sequential file. In order to avoid having to read the entire sequential file, a document directory may be referenced first. It consists of pointers that indicate the position at which a document is stored in the sequential file.

It is clear that, while an inverted file structure allows more rapid retrieval of information, it uses more storage than does a purely sequentially structured data base since, in fact, a sequential file is also required in order to retrieve the details of any stored document item. However, as indicated below, the sequential file shown in Figure 2 may be stored in a coded form which occupies less space than the sequential file shown in Figure 1.

This is possible because the file structure of Figure 2 includes a term dictionary of all the different terms present in the sequential data base, and so, instead of repeating these terms in the sequential file, it is sufficient to store instead a pointer to the position of the term dictionary. This

[†] Presented in the "Conference on Large Data Bases," sponsored by the NAS/NRC Committee on Chemical Information, National Academy of Sciences, May 22-23, 1974.