Computer-aided indexing with SPITBOL and TEXTFORM®

Ron and Eve Gardner

Eve Gardner is the sole proprietor of Gardner Indexing Service, Edmonton, Canada.

Entry selection for indexing is best done manually; sorting and formatting can be program-controlled. The authors' programs were written in SPITBOL and TEXTFORM® specifically to help the indexer with her work. (TEXTFORM® was developed at the University of Alberta.) Problems related to the lack of alphabetizing standards are discussed, and some solutions presented. The computers' internal character codes are briefly described. Computerized typesetting by the indexer eliminates the second proofreading cycle and produces an index of high quality in a short time. The indexer's experience of the computer is outlined.

When computers began to reach beyond mathematical calculations many information workers saw how these machines might be useful for information retrieval and indexing. Several techniques and programs were found to be more or less successful, depending upon application: for example, KWIC (Key Word In Context). A recent one, SPIndex, is described in an excellent paper by Joyce Duncan Falk[1]. Data storage and retrieval are more successfully computerized than indexing. The difficulty in fully automated computer-indexing lies in the computer's lack of judgement. Intuition is impossible and understanding of language extremely difficult to implement in software. Computer chess games now seem to display a form of human judgement, but chess is rigidly defined and so an algorithm at least seems possible.

This paper presents the problems set to a computer by 'back-of-the-book' indexing, followed by one possible set of solutions.

Indexing can be subdivided into three processes:
A) The choice of entries (or headings) for the index
B) The alphabetizing and arranging of the entries
C) The typesetting and printing of the completed index.

Choosing the entries

Index entries may be poorly selected by computer because, as mentioned already, its program lacks the necessary judgement and knowledge of language. A good index is not merely an ordered set of words[2] (a concordance) and phrases exactly duplicated from the text, although even this can be better than no index at all. Rather, the index should contain words and phrases describing in a *condensed* form what is to be found in the text *and* anticipating the needs of the reader or researcher. These chosen words and phrases are often quite different from their relatives in the text, even though they must all be related in language and subject. For example, in the index for *A distant mirror* by Barbara Tuchman we find the entry 'Berry, Jean, Duc de, . . . denounces Craon 413-14'.* Another example is taken from the index for *Peter the great* by Robert K. Massie: 'Affray of 1713 (Kalabalik) 592-6'.†

It will be a long time before any algorithm can accomplish such a task, so we believe the choosing of entries is best done by a person knowledgeable in the subject-matter of the book, and who understands what its readers will be looking for. Therefore this part of the process is done manually in our indexing system.

Sorting and formatting

The sorting and formatting of entries may not be well done manually, as indexers may find the work boring and tedious and be confused by large collections of words and numbers; the labour cost, too, is increasingly discouraging. However, the computer can do this very well—*provided* the task can be well and fully defined. Herein lies one problem; as all indexers are well aware, there is no universally acceptable standard for sorting or alphabetizing the entries of an index.

Even the basic decision as to whether to use 'word-by-word' or 'letter-by-letter' has yet to be made. Both approaches are widely used, and some studies have been carried out in an attempt to determine which is the better,[3] but so far no clear advantage has been found. *A manual of style*[4] strongly prefers the letter-by-letter method, but *British standard 1749:1969*[5] does not recommend it. The *Library of Congress* rules follow word-by-word alphabetizing, and the *Encyclopædia Britannica* follows letter-by-letter. Our program is able to use either; the editor decides. The program simply asks the operator 'Letter-by-letter sort?', to which she answers yes, or no.

*In fact the word *denounce* is not used in the text, but describes what de Berry said and did.

†The word *affray* is not used on any of these pages. The subject is also indexed under *Tumult*, which is used in the text on pages 592-3.

Some finer details now come to light. Upper-case letters are not the same as lower-case letters, and must be kept in some logical sequence. Punctuation may or may not be significant for sorting, depending upon the actual punctuation used. Numbers (both Arabic and Roman) must be correctly handled. In sorting, a page number must be treated differently from a number that represents, say, a year. Roman numerals should be sorted according to the *values* they represent: e.g., IV, V, IX, X, and not, IV, IX, V, X, as if they were alphabetized.

A great many computers have general sorting programs in their libraries, to be used either by the operator or by another program. These sorting programs vary from simple to complex, and of course produce results related to their degree of sophistication. Usually the sort program arranges the symbols of its input file according to the internally assigned binary value of the symbol. The size of the symbol table (or character set) varies from machine to machine, those of microcomputers being the smallest. (Mary Piggott[6] gives a good overview of the definitions of micro, mini, etc. computers.)

The value assigned to a symbol will be different in different machines, with micro and mini computers using ASCII code representation whose values range from 00 through 7F (hexadecimal). Large mainframes such as IBM use EBCDIC code with values ranging from 00 through FF. Thus there are 128 ASCII characters and 256 EBCDIC characters. For example, the symbols '0', '3', 'a', 'A', and SPACE have the ASCII values of 30, 33, 61, 41, and 20, respectively, but have the EBCDIC values of F0, F3, 81, C1, and 40, respectively (again hexadecimal notation). Binary and other number systems are discussed in nearly all standard textbooks on computers. If this group of 5 symbols were to be sorted according to their binary values, the resulting order would be SPACE, 0, 3, A, and a, for ASCII coding; but would be SPACE, a, A, 0, and 3, for EBCDIC coding. Of course not all sort programs are so simple-minded as this, but the collating sequence has been *predetermined* by the programmer of the sort routine and is usually not suitable for index sorting.

Reassigning character values

That last paragraph demonstrates the importance of reassigning values to the characters making up the entries of an index. A programming language called SPITBOL is an excellent choice for this job, because it was designed to process character strings. SPITBOL was developed at the Illinois Institute of Technology, and is nearly identical to its parent SNOBOL,[7] developed in the early 1960s by Bell Telephone Laboratories. The fundamental difference is that SPITBOL is a 'compiler' language, and thus executes much faster than SNOBOL, which is an 'interpreter' language. The term 'interpreter' means that every word of every statement in the program must first be translated by the computer into its native machine-language at execution time, *each* time the program runs. This of course greatly increases the computing time. The 'compiler' types are translated *once only* into machine-language, and the translated form is saved in a file for use each time thereafter.

```
*
* CONVERT ROMAN NUMERALS TO ARABIC
*
ROME        R = REVERSE(R)
NEXT        LD = TD
            R ANY('IVXLCDM') . TD = '' :F(DONE)
            TD 'I' = 1 :S(COMPARE)
            TD 'V' = 5 :S(COMPARE)
            TD 'X' = 10 :S(COMPARE)
            TD 'L' = 50 :S(COMPARE)
            TD 'C' = 100 :S(COMPARE)
            TD 'D' = 500 :S(COMPARE)
            TD 'M' = 1000 :S(COMPARE)
COMPARE     GE(TD,LD) :F(MINUS)
PLUS        A = A + TD :(NEXT)
MINUS       A = A - TD :(NEXT)
DONE        R = A
```

Figure 1. Part of the SPITBOL subroutine to convert Roman to Arabic.

The original program as written by the programmer is called a SOURCE FILE, and the translated form is called the OBJECT FILE. The translator programs are called compilers or interpreters, and are unique for a given machine-language. A number of indexers using BASIC in microcomputers have been reporting long execution times, primarily because BASIC is an interpreter type, but also because it is too general. BASIC is cheap and readily available, but SNOBOL and SPITBOL are not available for use in microprocessors and minicomputers—with the possible exception of some models of Digital Equipment Corporation, such as the PDP-10.

One of our programs (named simply INDEX) was written in SPITBOL to reassign character values quickly, replace words such as *St.* with *saint,* and delete articles *the, a,* and *an.* The INDEX program keeps these rearranged entries with the original entries until after sorting. INDEX then combines and formats only the original entries (which are now alphabetized), and adds some special commands for TEXTFORM® . For our file of 2099 lines for the *Civilizing the west*[8] index (see figure 3), the sorting time was 1·01 CPU seconds (the time required by the Central Processing Unit of the computer for this *specific* task. This is not *elapsed* time, which may be 10 seconds or more if the computer has many users at the time the task is executed). The rearranging and formatting took an additional 10·35 seconds of CPU time. The actual sorting was done by a utility program *SORT, called by INDEX from the computer's library.

The indexer values two things: time and accuracy. The keyboarding is done by the indexer with the help of

```
52      Macdonald, Sir John A.
        Gladstone, William E.
        Kimberley, Lord
        Lansdowne, Lord
        Manning, Cardinal
        Galt, Sir Alexander Tilloch\L2high commissioner in London
        Gillespie, Alexander
        Canada Company
        Brassey, Thomas
        Grand Trunk Railway
        Canadian Pacific Railway
        Homestead Act
        Hudson's Bay Company
        Stephen, Sir George
53      Galt, Sir Alexander Tilloch
        Macdonald, Sir John A.
        Stephen, Sir George
        Canadian Pacific Railway
        Dennis, John S., Sr.
        Brassey, Thomas
        Gillespie, Alexander
54      Macdonald, Sir John A.
        Galt, Sir Alexander Tilloch\L2in foreign affairs
        Stephen, Sir George
        Canadian Pacific Railway
        Gillespie, Alexander
        Brassey, Thomas
        Galt, Sir Alexander Tilloch\L2high commissioner in London
```

Figure 2. A section of input as typed at the video terminal, for processing by the INDEX program. Taken from our file for *Civilizing the west.*

another program called ENTRIES, which checks the entries for certain errors, changes abbreviations to complete entries, etc. Extra coding of entries during keyboarding is kept to an absolute minimum, to shorten time and increase accuracy. The entries are typed at the video terminal from a preliminary handwritten list, in columns. The left column carries the page numbers (each being typed once only), the right column the entries (see figure 2). A given entry must be duplicated each time it appears on a different page in the book.

Any abbreviation must have been previously defined to the program, otherwise an error message is given. For example, if 'Sir Alexander Tilloch Galt' is mentioned several times in the book, a definition is given to the program: $Galt = Galt, Sir Alexander Tilloch. Subentries up to 4 levels are indicated by the codes: /L2, /L3, etc. Exceptions to the normal sorting rules are handled by/SS, (Special Sort), but these are remarkably rare. These codes may be used in combinations. This same program provides access to the system EDITOR to make changes to the file, and also keeps a log of time, cost, filename, etc.

Our video terminal is connected via leased line to the main computer at the University of Alberta, running under MTS (Michigan Terminal System). The rate is comparable to prevailing commercial computer services, and operation is straightforward. The computer charge for producing the *Civilizing the west* index of 513 entries in typescript was about $95. When the terminal power is turned on, the screen asks for an ID (operator's identification) and a PASSWORD. If these are correctly entered, the user is then signed on. The command RUN ENTRIES calls up the ENTRIES program, which asks the operator for the name of the file to be used. (If this is the first time, then the file is automatically created.) Later, when the entries are complete, the command RUN INDEX calls up the INDEX program, which asks several questions, completes the sorting and formatting, and then calls the TEXTFORM[80] program to do the printing or typesetting. At this point minor changes and corrections can be made to the index by using the EDITOR and then rerunning TEXTFORM[80].

Printing the index

In recent years computers have been used more and more frequently in the printing industry to relieve the typesetter of mundane tasks. Typesetting an index is, in fact, easier than setting normal text, because the index has shorter lines, more blank spaces between groups of lines, and no figures or tables. This makes it easier to arrange the entries neatly on the page with all columns of equal length.

When the indexer produces the index in manuscript form for the printer, the entire document must be keyboarded again by the printer—with possible errors, further proofreading, and corrections. It is indeed a pity to have a correct and proper index in machine-readable form and not let the machine handle it.

Figure 3. A section of the completed index reprinted by permission from *Civilizing the west: The Galts and the development of western Canada* by A. A. den Otter, The University of Alberta Press, 1982.

If the index is to appear on the page in equal-length columns, then attention must be given to where the column ends. For example, a list of page numbers following a heading must not be broken at the foot of a column, because the reader might miss the remainder of the list in the next column. Similarly, a series of second-level entries should not be broken at the end of a page. Since the typesetter must maintain a fixed column length and yet not be allowed to end the column in certain places, how is he to do his job? The way round this impasse is to use a process referred to in printers' jargon as 'feathering' or 'leading', that is, adjusting the amount of white space between lines and between groups of headings to make the column end at the proper place. Needless to say, this *feathering* should not be noticeable to the reader. *Feathering* is easily done by a computer program *if* it knows where it may or may not end a column. It simply calculates the spaces required, according to a given set of rules.

The difficulty hinted at here, is in telling the program where it may end the column. Word-processors are not yet smart enough to *feather* type, but are continually improving. TEXTFORM® is a text formatter and has considerably more capability than any word-processor, mostly because TEXTFORM® runs in a large mainframe computer rather than a minicomputer. It handles most of the mechanics of producing a written document, such as justification, automatic hyphenation, changing type-styles, ordering and placing footnotes and quotations, and controlling a modern phototypesetting machine. The manuscript for this paper was produced with the help of TEXTFORM® . It is not yet able to *feather* type, but with further development perhaps some day it will.

Again, SPITBOL has come to our rescue. We have written a program in SPITBOL which avoids nearly all the 'bad breaks' (printers' jargon again) by telling TEXTFORM® what size of space to use *and* by *automatically* inserting 'continued' into the text, where required. (We have named this last program BREAKER.) TEXTFORM® now translates the finished index into streams of machine-language codes which only a specific machine can understand. After the phototypesetting machine completes its task, the finished 'Camera Ready Copy' is sent to the printer. Each page has only to be photographed, to make the master printing plate used in the press.

Some printers are now accepting machine-readable text from a magnetic tape or from a telephone line,[7] which then controls their phototypesetter. We have done this with a tape, but not yet with a telephone line.

An indexer's experience of the system

As a non-technical person using a new and highly technical system, I have certainly experienced frustration, but this is more than compensated by the pleasure and pride of turning out a creditable product. When we hit upon the idea of preparing indexes with the aid of a computer, both the programmer and I found the challenge of devising a completely new system irresistible.

Although I find typing on a terminal keyboard much easier than on a typewriter (corrections can be made so quickly and easily), I cannot think as well on screen as I can on paper, and therefore write all my entries in longhand on scratch pads. Beginning, if possible, with the printer's galleys, or even the manuscript, I sit in an upholstered armchair with the galleys and my scratch pad on a board resting on the arms. There I am comfortable enough to work for several hours together. The pages of the pad have a narrow column ruled at the left to accommodate page numbers and, later on, line numbers from the computer file if I should need them for reference.

On the first reading I simply write long lists of possible entries and note, on a separate sheet, errors and inconsistencies for the information of the editor. When the page proofs arrive I read again, delineating the book pages with coloured pencil on my pages and marking in page numbers. Unnecessary entries are scratched out at this stage, new ones added, and subheadings chosen.

Next I type the entries into the computer, and frustration arises from the work of my programmer, who is constantly improving our program; once he even revised it completely! I am somewhat mystified by the fact that I can practically memorize a 500-page manuscript, and know approximately where, among the galleys or page proofs, to search for a certain reference, but I do not understand the language or workings of a computer and cannot seem to remember what I am told. However, I persevere, always returning to try again after my anger has been vented elsewhere. I am repaid by the delight I feel each time a new improvement in the program works well and my typing time is reduced.

After proofreading and correcting a printed copy of the file as it was entered in the computer, I run INDEX, which combines and alphabetizes the entries and prints proof copies for myself, the editor, and the author to proofread. Corrections and alterations can still be made very quickly in the file at this point.

I have given much thought to ways of improving this system but it still seems the most satisfactory method. I can write entries by hand in long columns without giving thought to how I accomplish that task; my brain remains fully engaged with making decisions about the index. Alternatively, typing the entries directly into the computer demands that I concentrate on the instructions to the program (abreviations, second levels, italic type, etc.). I am not yet at ease with a terminal to the point of combining the two steps. Stopping to type my entries would certainly distract me from the subject-matter of the manuscript.

Also, I would be charged terminal connect time of $3.00 per hour while reading and making decisions if I were doing both steps together, and the many alterations I now do quickly by hand would cost more if done by the computer. It is faster and cheaper to strike out an unwanted heading with a pen than to type 'd 310' (delete line number 310). It is not economical in time or money to type entries into the computer until they read very nearly as the indexer wants them in the completed product; as it is, I find there is more than enough editing to be done after the first proof is run.

The final satisfaction comes when I am able to deliver to the publisher a finished index, ready to be printed as specified by the book designer, by the time that the editor and designer have finished their own work; they have not had to wait for weeks upon the indexer.

References

1. Falk, Joyce Duncan. Computer-assisted production of bibliographic databases in history. *The Indexer* **12** (3) Apr. 1981, 131-9.
2. Klumpner, George H. Disadvantages of computer-generated concordances. *The Indexer* **12** (4) Oct. 1981, 210-12.
3. Hartley, James; Davies, Lindsey; and Burnhill, Peter. Alphabetization in indexes: experimental studies. *The Indexer* **12** (3) Apr. 1981, 149-53.
4. The University of Chicago Press. *A manual of style.* 12th ed. Chicago and London: The University of Chicago Press, 1969, p. 415.
5. British standards institution. *British standard 1749:1969.* Specification for alphabetical arrangement and the filing order of numerals and symbols. p. 7.
6. Piggott, Mary. Do you process your words correctly? *The Indexer* **12** (4) Oct. 1981, p. 188.
7. Pasachoff, Jay M. and Kutner, Nancy P. Computer assistance in indexing with *INDEX. *The Indexer* **12** (4) Oct. 1981, 173-4.
8. den Otter, A. A. *Civilizing the west: the Galts and the development of western Canada.* Edmonton (Canada): University of Alberta Press, 1982.