Computational Approaches to Alphabetization and Routing in Phrasal Dictionaries

Richard A. Spears

This is a time of comprehensive technological advances in lexicography, due in large part to computers. Properly equipped microcomputers allow persons with interesting ideas to take part in or initiate advances in lexicography which use the new technology. Moreover, anything that minimizes costs is likely to make innovations appealing to publishers. Modern 16 or 32 bit microprocessors offer considerable computational power at low cost, and inexpensive high-capacity fixed-disks make rapid processing a reality on a microcomputer. Readily available computer software, especially shareware,[1] can encourage individuals to experiment and innovate.

This paper examines some of the special problems encountered in alphabetizing phrasal entry heads[2] and in routing users through phrasal dictionaries, particularly learner's dictionaries. A phrasal dictionary contains a predominance of entry heads which are multi-lexical rather than single words. Typical of phrasal dictionaries are Partridge (1977, 1978), Courtney (1983), Long (1979), and Spears (1987a). I will discuss specific ways of handling alphabetizing and routing problems on a microcomputer, with special reference to the conceptualization of a lexicographical workstation. Some of these techniques might be incorporated into a future lexicographical workstation of the type being discussed by the EURALEX Working Group on Dictionaries and Computers.

The techniques described here are for use with an inexpensive multi-window, programmable text editor and free or inexpensive utility software. Sophisticated computer programs which will do many of the things suggested here already exist or are in development, but these programs, due to their cost, their specialized hardware requirements, or their proprietary nature, may not be available to individuals who wish to experiment or design innovative lexical products.

Alphabetizing

Problems in alphabetizing are due to the complexity of the alphabetizing task itself and to the computer program used to alphabetize. The alphabetizing policy in phrasal dictionaries is more critical to the successful and efficient use of the dictionary than it is in normal, single word dictionaries. Of course, even the latter contain numerous multiword entry heads, but not enough to cause general problems. Where there is more than one word in an entry head, there are a number of different alphabetizing schemes which can determine the possible location of a specific entry. Some dictionaries use mixed criteria. For instance, the AMERICAN HERITAGE DICTIONARY (AHD) uses "strict" or absolute alphabetizing except in the case of biographical entries such as **Jones, John Q.** where everything after the comma is ignored in alphabetizing. Dictionaries differ as to whether terms consisting of an abbreviated title followed by a name, such as **St John**, will be found at the title or the

name. That is, will **St. John** be found at "j," "sa," or "st"? Eric Partridge favored an order where the blank space comes before the letter "a," with the hyphen counted the same as a blank space. This order puts *cotton-wool* before *cottonocracy*. Anyone who is unfamiliar with this system may give up the search for *cottonocracy* after reaching *cotton-wool*. A variant of absolute alphabetizing ignores hyphens, spaces, and punctuation and allows a consistency which is very easy to explain to users. Absolute alphabetizing also eliminates the problems caused by the user's lack of knowledge about whether an entry head is written solid as one word, hyphenated, or as two words as with *onetwo, one-two,* or *one two*. Ideally, the policy on alphabetizing should be simple, unambiguous, and easy to communicate to the user who may be a learner with only an elemental knowledge of the language of the dictionary. Such users may lack the requisite comprehension skills to understand an explanation of a complex policy on alphabetical order.

A major problem in editing phrasal dictionaries is deciding what each entry head should be. There are few "fixed phrases" which cannot be varied in some way. In addition to determining what the entry head is to be, one must make certain that users can find it. Various schemes have been employed to provide users a means of reaching a "main" or "key" word in a phrase. These schemes often have drawbacks, primarily where the "main" word—as perceived by the user—belongs to an open-ended class of lexical items which is in fact a variable in the phrase, rather than one of the invariant components considered to be the "main" word by the lexicographer. This problem and its solution is discussed in Spears (1987b).

The alphabetizing of phrasal entry heads offers special problems because of the limitations imposed by computer sorting sequences. Simple sorters cannot be controlled to ignore nonalphabetic characters such as space, hyphen, comma, period, and other punctuation, and many of them will sort on only the first 10 to 40 characters of a line. Many fast and flexible sorters are limited by the size of files they can sort. Even capital letters can upset one's alphabetical order, although most sorters can be adjusted to ignore the difference between upper and lower case. A related problem in all lexicographical work is the need to sequence not just a list of words or phrases, but the equivalent of paragraphs of various lengths—that is, many entry blocks[3] of various lengths. There are programs available commercially, the so-called "garbage bag" database programs, which are supposed to sort paragraphs into alphabetical order, but—like other sorters—they do not allow any control over the ordering of nonalphabetic characters. Some commercial sorters are not reliable. I have used two commercial paragraph sorters which lost entire entries in the sorting process. Both conscientiously filled in the lost space with parts of other entries so that the loss was not evident in the size of the files. Database programs, such as dBase III, will not handle large entry blocks and offer little control over sorting. Some sorters such as Supersort[4] will permit sorts in any conceivable order, but there is no turnkey sorter known to me which allows one to ignore totally the code for a space, hyphen, comma, or other punctuation.

A shareware program called Qsort[5] has proved to be very useful for dictionary work. This program is extraordinarily fast and can sort files and records of any size, limited only by disk storage space. In conjunction with Qsort, a programmable editor such as Vedit Plus[6] can be used to exercise *total* control over sorting by creating a "pseudo entry head" edited to favor or ignore any non-alphabetic characters. A short macro program operating in one window of this editor locates

the entry head and copies it to another window of the editor where the macro deletes the characters which are to be ignored. Typically, all spaces, hyphens, and other punctuation will be deleted. Where an entry head includes an element which is not to be alphabetized, such as (SOMETHING) in the string **put** (SOMETHING) **into action,** the entire element (SOMETHING) can be deleted leaving only **putintoaction.** The abbreviations "St." and "Mr." can be spelled out as "saint" and "mister" for sorting purposes. The macro can find sequences like **Jones, John Q.** and delete the comma and whatever follows it. The pseudo entry head is then returned to the main editor window and placed above (or ahead of) the normal entry head. Here are examples of possible deletions or changes.

Normal Entry Head	Pseudo Entry Head
Make my day!	Makemyday
Keep a stiff upper lip.	Keepastiffupperlip
St. Elmo's fire	saintElmosfire
bright-eyed and bushy-tailed	brighteyedandbushytailed
put (something) into action	putintoaction
Cool, man, cool!	Coolmancool
Jones, John Q.	Jones

With the pseudo entry heads in place above the normal entry heads, the program calls Qsort to sort all the entry blocks on the pseudo entry head, arranging the entry blocks into absolute alphabetical order. Finally, the macro deletes the pseudo entry heads. A file comparison program is needed to make certain that the unsorted and sorted versions of the files contain the same material. File comparison programs which rely on CRC [cyclical redundancy check][7] will not work. Other file comparison programs, including the DOS COMP program produce enormous quantities of useless error listings in this kind of task. A simple batch file which uses the DOS FIND command can count the occurrences of specific characters before and after sorting. If the file's name is COMPARE.BAT, the command line is

COMPARE UNSORTED.DOC SORTED.DOC

The assumption is that the two files UNSORTED.DOC and SORTED.DOC will have the same number of e's, t's, periods, and backslashes. COMPARE counts the occurrences of specified characters in the before and after versions and puts the results in the file named SCORE. The contents of COMPARE.BAT are:

```
find /c "e" %1 >score
find /c "e" %2 >>score
find /c "t" %1 >>score
find /c "t" %2 >>score
find /c "." %1 >>score
find /c "." %2 >>score
find /c "\" %1 >>score
find /c "\" %2 >>score
type score
```

A display of the file SCORE which looks like the following, indicates that there have been no changes in the files other than re-alphabetizing. This shows that both files have the same number of e's, for instance, no matter where in the two files the

e's might be. If the counts in each pair of scores match, the files are assumed to be undamaged.

unsorted.doc: 189
sorted.doc: 189

unsorted.doc: 188
sorted.doc: 188

unsorted.doc: 103
sorted.doc: 103

unsorted.doc: 31
sorted.doc: 31

More elements of comparison can be added to make the program even more effective. Such a checker should be run after sorting of a file.

Routing

The term routing, as used here, refers to the use of devices other than alphabetizing to direct users to specific locations within the dictionary. These devices are generally referred to as cross-referencing which is a misleading term since very little of cross-referencing is actually mutual or "cross." Phrasal dictionaries—because of the variability of phrases—may make extensive use of routing devices. A learner's dictionary, may also require more routing than a standard desk dictionary.

The traditional cross-referencing devices, such as "see," *cf.*, and the use of small caps do not always indicate what a user is expected to do or what might be found at the reference. Printing an expression in small caps may indicate only that the expression occurs as an entry head at some other place in the dictionary. "See X" may actually mean "find what you are looking for at X or within X," but this instruction may also refer the user to vaguely related material which is of little immediate use. Instructions as to what these terms mean are seldom very specific in dictionaries. The abbreviation *Cf.*, from Latin *conferre*, is always defined as compare—although another sense of *conferre*, "to bring together," is probably a more accurate translation.

It is suggested that—rather than elaborating on the traditional cross-referencing terms—routing schemes are better structured along the lines of the flow control concepts of computer programming. Notions of position and movement in a computer program as expressed by BASE, JUMP, JOIN, and READ, for example, are better than the ambiguous *cf.* or see. A BASE entry block would be the current entry block being considered by a user. In a JUMP the user would abandon the current BASE and move to another entry block which would then become the new BASE entry block. In the case of a JOIN the user would move through a series of entry blocks before returning to the BASE entry block. A READ would lead the user to another entry block only for specific information and then return the user to the BASE entry block. These positions and movements are useful in *planning* and *describing* routing policies in a dictionary. I am not suggesting that these terms be used or even that these are all useful activities in routing. Indeed, some of these

ought to be avoided. For instance, one might wish to avoid the JOIN altogether, there being no good way to indicate where a user should return to the BASE entry block. A policy might be formulated where the READ activity is replaced with a listing of the needed material right in the BASE entry block.

It is also in the checking and verification of routing that microcomputers can be of great help. Checking a dictionary manuscript to make sure that the elements users are led to are actually there in the proper form can be handled easily with the help of a programmable editor and other software. A macro in the editor extracts the entry heads and places them in file A. It then extracts all the routing targets and places them in file B. File B is searched and each routing target which matches an entry head in file A is deleted. Only the mismatches remain in file B and must be corrected. Again speed and storage capacity make this possible. A utility program such as FGREP[8] which allows speedy searches is essential to seeking the location of the elements which play a role in routing. In a heavily cross-referenced dictionary, any tampering with the user targets [entry heads, sense numbers, etc.] must be followed by a check of the entire dictionary to see if the altered form is a target for any routing directions. What is really important is that editors of learner's and other specialized dictionaries recognize that routing is an important policy matter needing special attention, and that the traditional approaches to cross-referencing—while adequate for desk dictionaries—do not provide enough options for some kinds of specialized dictionaries.

Summary

Alphabetizing—though seemingly simple—is a fairly complicated matter with many variations. The tedious procedure of alphabetizing is one which begs for the help of a computer, but attempts to bring alphabetizing under control are frustrated by standard computer alphabetizing programs. Certain types of reference works, learner's dictionaries in particular, may require a very simple and consistent policy which is easy to communicate to the user. A lexicographical workstation should include very flexible alphabetizing schemes or provide immediate access to programs allowing maximum flexibility. The use of pseudo entry heads for sorting offers this flexibility.

Aspects of routing in general—usually coming under the heading of cross-referencing—should be quite specific about what a user is meant to do. The usefulness and efficiency of a reference work can be maximized through the careful d esign of routing. At a minimum, there should be a distinction made as to whether the user is meant to call in information from another entry or abandon the current entry and go to a different one. Some indication of the importance of the referenced information would be helpful. The traditional cross-referencing devices represented by small caps, "see," and *cf.* are not specific enough for signalling different types of routing. The routing used in computer program design could provide a good model for dictionary routing schemes.

It is clear that lexicography and computers are wedded forever. Different ways of using the computer effectively are still being devised; the turnkey lexicographical workstation being only one approach. A possible disadvantage of a turnkey system is that its designers must anticipate everything that someone might want to do, or

make it easy to expand or revise the program. The more complex and flexible such a workstation is, the more it places itself beyond the reach of the average researcher either because of the cost of development or the complexity of its operation. An alternative approach is to utilize existing commercial programs supplemented with public domain and shareware programs. Shareware is generally inexpensive and often of very high quality. Many shareware authors are eager to receive suggestions which will improve their products, and some authors are willing to customize their software to the user's needs. The proper use of shareware is like having a team of programmers working on the various modules of one's lexicographical workstation. A programmable editor becomes the central organizing feature of this kind of workstation. One supplements it with a collection of free or inexpensive software from which one can select and master only what is required.

Dictionaries do not have to conform to or be limited to a single official notion of what a dictionary is. Dictionaries are reference books designed to meet specific needs. We are increasingly recognizing new needs and have at our disposal computational means to satisfy newly discovered needs. In addition to reducing costs and increasing efficiency, the technology available through computer use may lead us into new and better ways of serving our fellow humans.

Notes

[1] Shareware refers to computer software written by both amateurs and professionals and distributed by computer bulletin boards and similar services such as CompuServe. You are urged to send a sum of money to the author of the program if you like the program and you use it. Shareware programs tend to be highly specialized, and many are as well written and efficient as commercial programs. Some shareware programs are available only as demonstration versions and one is obliged to buy the full version of the program. Most range between U.S. $5.00 and $35.00.

[2] "Entry head" was proposed by the DSNA Committee on Lexicographical Terminology as a generic term for *definiendum*, covering "lemma," "compound," or "phrase," rather than *head word*.

[3] "Entry block" is used in the AMERICAN HERITAGE DICTIONARY "Guide to the Use of the Dictionary." It includes an entry head and everything else up to the next entry head.

[4] Supersort is a fast and flexible commercial program with a restriction on the length of records which limits its usefulness in lexicographical work. It was originally released for the CP/M operating system by MicroPro International in 1978. An MS-DOS version is available in the U.S.

[5] Entire entry blocks are sorted reliably by the shareware program Qsort. There is a special setting for what is called a "lexical sort." In this mode, all sorting is done without regard to case, except that where there are identical strings aside from case, the upper case version comes first. Qsort was written by Ben Baker, RR □1, Box 637, East Alton, IL 62024, U.S.A., who asks U.S. $20.00 for the program.

[6] Vedit Plus is a commercial program produced by Compuview Products, Inc., 1955 Pauline Blvd., Ann Arbor, MI 48103, U.S.A. They will send a free demonstration disk on request. This program is a text editor and requires an additional text formatter to be used as a word-processor. Vedit plus is a multiple window editor which allows up to 37 versions of the editor to operate at once, each in a separately accessible window. A programing language is included which allows highly complex and sophisticated text manipulation programs to be written. I have used Vedit Plus to make concordances and sort entry blocks of very long

files. The program also has a visual mode which can be configured to match any word-processor.

[7] The Cyclical Redundancy Check [CRC] provides a parameter which can be used to compare files. Absolutely identical files should have the same CRC figure. A comparison of a sorted file with its original unsorted version will rarely if ever produce the same CRC.

[8] FGREP is a public domain program which will search a 2 megabyte file in under twelve seconds for a string and record the location of each occurrence in a separate file. The program is from Cove Software Group, P.O. Box 1072, Columbia, Maryland 21044, U.S.A., and is available at no cost.

References

Cited Dictionaries

Courtney, R. 1983. LONGMAN DICTIONARY OF PHRASAL VERBS. Harlow: Longman.

Long, T.H. (ed.). 1979. LONGMAN DICTIONARY OF ENGLISH IDIOMS. Harlow: Longman.

Morris, W. and M. Berube (eds.). 1969/82. THE AMERICAN HERITAGE DICTIONARY. Boston: Houghton Mifflin.

Partridge, E. 1977. DICTIONARY OF CATCH PHRASES. New York: Stein and Day.

Partridge, E. 1978. A DICTIONARY OF CLICHÉS. (5th ed.) London: Routledge and Kegan Paul.

Spears, R.A. 1987a. NTC'S AMERICAN IDIOMS DICTIONARY. Lincolnwood, Illinois: NTC Publishing.

Other Literature

Spears, R.A. 1987b. 'Managing Phrasal Data With a Microcomputer Generated Concordance' in *The Sixth International Conference on Methods in Dialectology*. Bangor, Wales.