Complex Data, Simple Tools: An Introduction to Text Retrieval Packages*

by Andrew Marchant-Shapiro [1]
Departments of Political Science and Sociology
Union College, Schenectady, N.Y.

Introduction

Before the advent of personal computers, qualitative researchers could often be found "**waste**" deep in typed interview transcripts. Dealing with these transcripts often meant hours of searching for a particular passage or pattern. For many, little has changed — except that the transcripts are now word-processed instead of typewritten. But precisely because they *are* word-processed, these transcripts open up new possibilities for computer-aided retrieval and analysis.

This article provides an overview of one class of software programs, text retrieval packages (TRPs), that can provide significant assistance to qualitative sociologists with minimal investments of both time and money. Using a hypothetical text retrieval package, I suggest some techniques that sociologists can use to maximize the utility of TRPs. I outline the basic characteristics of TRPs, and describe a few commonly available software packages that present variations on the TRP theme. The techniques introduced here are not specific to any one system, and may be used to advantage with a wide variety of text retrieval packages.

I should make clear at the outset that this article is about improving access to textual data, with specific application to qualitative data such as transcripts from unstructured ("conversational") interviews. None of the TRPs described in the following pages provides for *analysis* of qualitative data. While such programs are readily available, and are used by a growing number of qualitative analysts, they are not my concern in this article. Rather, the programs discussed below are simple tools that provide qualitative researchers with greatly improved access to their complex data.

Thinking of the Interview as a Database

When a sociologist hears the word "database," he or she is likely to think of a collection of coded data arrayed in rows (cases or 'records') and columns (variables). Normally, such databases are manipulated with software programs known as data base management systems (DBMS). The DBMS makes it possible for the researcher to gain rapid access to specific sections of his or her data. For example, a researcher using DBASE IV, a popular DBMS program, might want to see the ages of all those individuals in her database who lived in Illinois in 1970. Depending on how the data are structured, she might give the following command:

list age for "IL"$state_1970

The DBMS program would first locate those rows of data for which the variable STATE_1970 had the value "IL," and then isolate the variable AGE in each such record and print its value on the screen. A typical output might look like this:

27
28
19
41
30

Note that by using the DBMS program, the quantitative researcher has gained great power in interrogating her data. She no longer needs to manually search for each case in which a subject was living in Illinois in 1970. This ability to isolate particular records for inspection is one of the reasons that DBMS programs have gained popularity with quantitative researchers. On large surveys, such programs are often used to ease cleaning of data and allow for the isolation and closer inspection of outlying cases.

The usefulness of similar strategies should not be lost on the qualitative researcher. There are many instances in which it is desirable to move rapidly to a section of an structured or unstructured interview that is marked by the occurrence of one or more key words or phrases. These range from the early days of a research project, when one is exploring the transcripts of recent interviews, to the final stages, when a researcher may need to find one or more quotations to reinforce her point. One may even want to test the notion that two words or phrases verbalizing particular concepts occur only (or most frequently) in conjunction with one another. In a set of interview transcripts that can run to hundreds of pages and millions of words, how can you find the particular passage, or passages?

One approach, to be recommended for its economy and simplicity, is to use the search function of your word processing program to look for the text in question. But

with a few exceptions (noted below), this limits you to searching for a single word or phrase in a single text file at a time. Moreover, more complicated searches (such as searching for all paragraphs of text that do *not* contain a particular word or phrase or combination of words and phrases) are beyond the capabilities of word processing programs.

This is where text retrieval programs come into play. The grandparent of modern TRP programs is a widely available program called GREP.[2] It originated on UNIX mainframes, and is today available on all computers that run the UNIX operating system. Moreover, various public domain[3] versions of the GREP program are available for most microcomputers, and a limited version of GREP, called FIND, is distributed by Microsoft with every copy of MS-DOS.

GREP is a simple program, but extremely powerful. In essence, you give GREP a word or phrase to search for, and it compares each line in a data file with that specific word or phrase. Lines that match can be counted, printed to the screen, or saved into a new file for further manipulation (alternatively, you can do the same thing for lines that *don't* match). A researcher might be interested in seeing how often the word 'credibility' appears in a particular interview transcript. The transcript is stored in the file TRANS017.TXT, so our researcher invokes GREP this way:

```
grep -n credibility trans017.txt
```

GREP scans each line of TRANS017.TXT for the pattern of letters forming the word *credibility*. The '-n' in the command causes GREP to number the lines in the file as it scans them. When it finds a line containing the pattern *credibility*, it prints that line to the screen, forming an output like this:

```
0023 and that was a serious problem for our credibility
0040 was it a credibility problem? No, credibility was
0215 Credibility. Plain and simple. If it hadn't of
```

The researcher now knows how often the word appears in the transcript, as well as where it appears. Since GREP works very quickly, a few such searches can give the researcher significant insight into his data in a very short time.[4]

Note, however, that there are some important drawbacks to the way that GREP handles the data. First of all, what we have retrieved are *lines* of the file, not sentences. From a computer's point of view, lines are a sensible units to use because it is easy to tell where one line ends and the next begins. The computer treats each line as an independent *record*. But from a human perspective, lines are not very useful as records: they may contain anything from a few words to a few short sentences, and they do little or nothing to establish the *context* within which any particular datum is found. GREP can rapidly locate all *lines* of text containing matching patterns; we need programs that can retrieve entire *chunks* of text (sentences or paragraphs, for example), context and all. At a minimum, we need to locate not only the statement containing the word or phrase for which we are searching, but also the stimulus that evoked that statement.

A second problem is that GREP is limited to searching for a single word or phrase at a time. While a skilled user can compensate somewhat for this limitation through the use of complex 'regular expressions' or root searches (see below), GREP is incapable of searching for phrases that break over lines and cannot examine text chunks for the presence and/or absence of multiple words and phrases. GREP is limited to searching for a single word or phrase (a sequence of words); we need programs capable of looking for combinations of words and phrases that may or may not be sequential.

Modern TRPs answer both of these needs and more. Rather than operate on single lines of text, they can operate on paragraph-sized chunks[5] and can make use of Boolean operators (see below) to allow for a variety of ways to combine search texts. Moreover, unlike word processing programs, TRPs can search many files with a single command, greatly speeding up the retrieval process.

In themselves, these improvements over word processors and GREP make TRPs powerful if simple-minded tools. To get optimum performance from a TRP, however, requires more than just aiming the program at a file and telling it to go to work. By adding a modicum of structure to the transcript of an unstructured (or structured) interview, we can realize significant benefits. Structuring a transcript actually involves three different aspects: *structuring*, in the sense of organizing a conversation into meaningful 'chunks'; *identifying concepts*, adding keywords to the record that either amplify the content of the conversation or actually represent analytic categories; and *problem prevention*, a process analogous to the 'cleaning' of quantitative data.

Structuring the Transcript

If the chunks of data that we wish to retrieve are larger than a single line, we need to structure the text so that the TRP we are using understands where a particular chunk begins and ends. How we structure chunks (or *records*) depends on the particular data we are looking at and how we plan to use it.

Consider the unstructured interview. Such an interview is a conversation, typically made up of paragraphs; the

first party speaks, then the second, then the first, and so on. Typically, the interviewer asks a question, then the informant replies, as in Figure 1. [6]

Figure 1

Q: Were you particularly worried about extremists coming into the movement at that time (1978)?

A: Not especially, no. Not until we got word that the news media had mixed up some of our group in the west with the Posse Committatus. That cut into our credibility something fierce, and made it very difficult for us to get sympathetic press coverage.

The question and answer — sometimes with followup questions and answers or other interactions — provide the context within which to understand a particular statement. If we can treat each question-and-answer set as a *record*, that is, as an independent datum, then we are well on the way to transforming what may be a long (and sometimes rambling) interview into a useful database. Within each record, we should have not only the full text of the answer, but the stimulus that evoked that answer. Bear in mind, however, that the real challenge is not understanding for ourselves what constitutes a record, but organizing our data in some way so that the computer's notion of what constitutes a record is identical to our own. Once we have arrived at a definition of a record that is adequate for our own use, we have to think about the structure of the records as the computer sees them.

From a computer's perspective, the most useful form of raw data is a file that consists of text (letters, numbers, punctuation and spaces) and a few special characters, such as carriage returns and line feeds. Such a file is known as an ASCII (American Standard Code for Information Interchange) text file, and uses characters in a standardized fashion.[7]

To divide a file into records that both the researcher and the computer/TRP will understand, use single spacing within paragraphs, and double spacing between paragraphs, as in the following example:

```
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX  <-record 1
XXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXX          <-record 2
XXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXX  <-record 3
XXXXXXXXXXXXXXXXX X
```

In this way, each paragraph of text becomes a distinct record, and the TRP can easily distinguish where one ends and the next begins.

Records should include, at a minimum, a question and the response it invokes. These should be divided in some way, however, so that we can tell at a glance at which part of the interaction we are looking. One way to divide between the two is to insert a line of hyphens (-) between question and answer. The one way *not* to divide the question and answer is with a double carriage return; this will make the question and answer appear to the TRP as two independent records.

Identifying Concepts

Not infrequently a conversation has more meanings than would be apparent from the text of the interaction itself. In such instances, we may wish to add still another section to the record — again, divided by a string of hyphens or other special characters — that consists only of *keywords* or comments pertaining to the interaction. Such keywords may simply clarify the meaning of the text of the conversation, or they may be analytical categories you have assigned to the particular record.

If you do use keywords, it is a good idea to use some special character to mark them so that the TRP can differentiate keywords from the rest of the text. For example, all keywords might be preceded and followed by the '*' character — *aggression*, *money*, and so forth. This helps to avoid confusion between words that are part of the text *per se* and others that are introduced by the researcher once the interview has been completed.

Preventing (and Resolving) Common Problems

While we are busily creating the perfect data record, however, we need to be aware of complications that we can introduce that may make searching difficult or impossible. The major problem is one that afflicts all text processing: misspelling and inconsistent spelling. This is a particularly nasty problem if someone other than the interviewer transcribes the interview. Computers are powerful but inflexible creatures; if you search for the name *Kamin*, you may find that the name doesn't come up in the database — because it has been entered variously as *Kemin, Kammin, Camin* and/or *Camyn*. There are flexible TRPs (discussed below) that may find one or more of these misspellings through the use of 'fuzzy' search criteria, [8] but it is probably best not to rely on technology to fix this problem after the fact.

The most straightforward answer to this problem is the spelling checker. These programs, often integrated with word processing programs, scan the text, either while it is being entered or once the file is complete, and locate words that do not match a dictionary file. Most spelling checkers have provision for an auxiliary dictionary,

which contains words not in the main dictionary but which are used by the writer. This is the place to establish a list of names of persons and organizations, so that misspellings will be detected at once and corrected. You should also supply yourself, or the person transcribing the interview, with a list of names and special terms that appear in the interview(s) with which you are working. As you proofread the transcripts, you can add to this list (and the spelling checker list) as you go along.

Sometimes, however, a new name or term comes up, or a name is garbled. You should have provisions for alternative spellings, such as the following:

after that [Mankoff/Mankov (0213)] told me that

An early step in going through the interviews could be to search for the '[' character, so as to locate and resolve problems. Alternatively, you can leave the variant spellings in place, in case context later allows you to interpret the garbled passage. If you are not transcribing the interview yourself, have the person who is insert the digit counter value for the point on the tape where the garble occurs. In this way, even imperfect transcriptions (and there are few perfect ones) will be useable by computer search programs.

It may sound like a lot of work to structure text in this way, but it is not really that difficult if the structuring is done when the data are first entered into a word processing program. If the researcher himself or herself is entering the interview, then keywords can even be added at the same time. The additional labor imposed by a simple data structure is a small price to pay for the ease of access that will result. In the next section, we turn to the issue of access in order to get a sense of what can be accomplished. Whether coded or not, once the data have been structured, the hard part is over.

Searching for Data

If we think of our interview data as now consisting of paragraph-sized records, each record consisting of a question and its associated answer and constituting a context for the statements therein, we are in a position to consider how we would like to specify which records to retrieve. Searches may be simple (for one word or phrase) or complex (for various combinations of words and/or phrases). If we have added keywords to the interview data, we may search for these as well.

Simple Searches

Recall the example of the quantitative researcher. Her search began by specifying a subset of possible records — those records that contained the value 'IL' in the variable STATE_70. The qualitative researcher does not have variables to work with in the same sense; instead, he has a chain of verbalized (and perhaps coded) concepts. While these are not consistent from record to record — only a few members of the set of all possible concepts are present in any given record — we *can* test for the presence or absence of particular words.[9] Consider Figure 2. This is the same paragraph shown in Figure 1, but now structured as a record and stored, with other records, in the ASCII file INTRVW.001:

Figure 2

Q: Were you particularly worried about extremists coming into the movement at that time (1978)?

A: Not especially, no. Not until we got word that the news media had mixed up some of our group in the west with the Posse Committatus. That cut into our credibility something fierce, and made it very difficult for us to get sympathetic press coverage.

---

extremist *posse* *media* *1978* *west* *perception*

This record contains a stimulus, a response, and a set of keywords that both overlaps (e.g., *media*) and categorizes (e.g., *perception*) the information contained in the interaction. The record shows the presence of such concepts as *extremist, movement, media, credibility, 1978*, and so on. On the other hand, it does *not* contain terms indicating such concepts as *electoral politics, formal organization*, or *legitimacy* (to name just a few possibilities). So, if we wanted to retrieve only those records that included a verbalized or keyworded conception of credibility, we might give a hypothetical TRP a command like this:

list 'credibility' in INTRVW.001

The result would be a listing of all of the records in INTRVW.001 that contain the term *credibility*, including, of course, the record shown above. If we wanted to search for all records that contained the concept *electoral politics*, the TRP would not retrieve this record. Conversely, if we searched for all records that did not refer to electoral politics, this record would be among those retrieved.

But suppose that we wanted to search more broadly — for variations on *credibility*. Suppose that our informant didn't actually use the word *credibility*, but said something like 'it was hard for us to be *credible*.' Since *credible* is not the same pattern of letters as *credibility*, the computer would not have found that record. But we can modify the search in one of two ways so that we are more likely to find appropriate records. We can either search using roots, or we can search using multiple terms.

If we are searching for variants on a single term, searching for a root can do the job. For example, we might search for the pattern common to both words, i.e., *credib*. To do a root search, consider all of the similar terms you want to retrieve and search for the common portion of those words. *Legitimacy*, *legitimate*, and *legitimation*, as well as variations such as *illegitimate*, can all be retrieved through a common root. If you use this approach, though, be careful not to shorten the root too much; if you do, the search results may be useless because large numbers of records containing 'noise' words satisfy the search.

Some TRPs allow for a variation on the root search method using *wildcards*. These are special characters that can be inserted into a search that will match any other character or combination of characters. If '+' matches any single character and '_' matches any group of characters, then 'gr+w' matches words such as *grew* and *grow*, and 'im_le' would match anything from *stimulent* to *impossible*. Obviously, wild card searches are also subject to 'noise' problems, and should be undertaken with care.

Complex Searches

While searching for a single word or for variations on a single word can be helpful in plowing through long transcripts, it is often more useful and more interesting to be able to choose records based on the presence of two terms, or on the presence of one and the absence of another. *Boolean operators* are ways of specifying logical connections between words and/or phrases. On-line information services such as Lockheed's DIALOG service make use of these operators, as do more common, PC-based systems such as WilsonDisc, and virtually all DBMS programs. The basic Boolean operators, AND, OR, and NOT, can be used singly or in combinations to set exacting criteria that records must meet before the TRP will retrieve them.

Widening the Search: Logical OR A root search works by using a single, less rigorous criterion for matches. In contrast, a multiple term search expands the search pattern by allowing a record to be retrieved if it satisfies one or more elements of a *set* of criteria. To construct such a set, we use the Boolean logical operator OR. For example, if we give a command to our TRP to:

list for 'credibility' OR 'credible' in INTRVW.001

A record that contains either word will be retrieved. Obviously this technique can be expanded so that concepts that may be expressed in a variety of ways can be searched. We might want to search for terms like 'credibility' OR 'legitimacy,' for example. Using the logical operator OR always widens the search, since a record that satisfies any part of the expression that has been ORed together is retrieved. Sometimes, however, ORing things together gets us more than we want. It is then that we can use another logical operator to tighten our search criteria.

Narrowing the Search: Logical AND and Logical NOT When we AND things together, we are telling the computer to retrieve only records that meet multiple criteria. For example, we could exclude the example record shown in Figure 2 from a search by asking our TRP for the following:

list for 'credibility' AND 'organization' in INTRVW.001

The record satisfies one criterion but not the other, so it is not retrieved. Only those records will be found that contain both words. Using AND takes care, because it is possible to quickly reduce the number of records that match the search to zero.

The utility of AND and OR is increased by adding the third logical operator, NOT. NOT allows the TRP to retrieve a record only if a particular term is not present in the record. NOT is seldom useful alone, but in combination with AND and OR, it allows for very precise specification of searches. If we wish to find only those records that refer to 'this', but not those that also refer to 'that', then we can search for 'this' AND NOT 'that'.

Grouping Logical Operators with Parentheses While many searches are easy to specify with one or two logical operators, searches can become quite complex, and it is important to specify the priority in which logical operators act. Fortunately, most TRPs allow the use of parentheses, which allow the researcher to specify the order in which the TRP evaluates logical relationships. We can develop searches such as ('credibility' OR 'legitimacy') AND NOT 'organization'. This particular search would first retrieve the subset of all records in which either 'credibility' or 'legitimacy' were present, and then reject the sub-subset of records which also contained the term 'organization'. If we had instead defined the search 'credibility' OR ('legitimacy' AND NOT 'organization'), the TRP would first find all records containing 'legitimacy' but not 'organization', and then retrieve as well all records containing 'credibility' *regardless of whether or not they included 'organization'*.

An example of the logical operators' power to differentiate among records may be in order here. Consider the following one-line records:

1. Then Bob told Carol and Ted.

2. But of course Alice and Carol told Bob and Ted.

3. Alice and Ted were outraged at that.

4. Finally, Alice left with Ferdinand.

Below are some search criteria and the numbers of the records that each search would retrieve. These should demonstrate clearly the different behaviors of the various operators.

'Bob' OR 'Carol' OR 'Ted' OR 'Alice'
(1,2,3,4)

'Bob' AND 'Carol' AND 'Ted' AND 'Alice'
(2)

'Alice' AND NOT ('Bob' OR 'Carol' OR 'Ted')
(4)

'Alice' AND ('Bob' OR 'Ferdinand')
(2,4)

Searching Using Keywords
With the use of Boolean operators, keywords take on a special significance. They are more than merely additional tags that we can use when our informants use varying terms to discuss a single concept. Through the use of AND, OR, and NOT, we can examine the relationships that exist between keywords that indicate coded concepts and the content of the conversation itself.

Recall that keywords are marked with special characters (*). These markers affect searching in particular ways. For example, a search on the term 'legitimacy' will be satisfied whether the term occurs in the text or in the keyword section. But '*legitimacy*' will *only* be satisfied by the term in the keyword section. By combining keywords and logical operators, we can do searches like this:

list for '*media*' AND 'credibility' in INTRVW.001

This search would find only those records noted and marked by the researcher as having some bearing on media issues, and then only the subset of these records that had verbal and/or keyword relations to credibility. By adding keywords to our records, we begin to approach the same kind of specificity and power in searching that DBMS programs afford quantitative researchers.

Making Use of the Output

The goal of all these manipulations is, of course, to find specific records within a large body of information. What you do with that information once you find it is up to you, but you should be aware that not all TRP programs allow you to save the data that you find. Some merely allow you to view the records that the program has retrieved. Most have provisions for saving some or all of the retrieved records to an ASCII file. Some, such as *Golden Retriever* (reviewed below), take you to the point in your transcript where the match occurred and allow you to save as much or as little of the surrounding material as you desire.

Since the usefulness of TRP programs lies in their ability to winnow data, as it were, you should probably avoid programs that do not allow you to save output to a new file. For example, *SeekEasy*, one of the programs reviewed below, has no provision for placing the retrieved text into a new file. This limits its usefulness in anything other than exploratory research, since the only way to record the results of your search is with pencil and paper (or the print screen key).

Once you have an output file, you can do several things with it. You can simply include the file, or an edited version, in a paper or article you are working on. Or, if the number of records retrieved is large, you may be able to treat the new file as a second-order database — searching more specifically within the file.

In any event, you should *always* take a look at the output file before including it in other documents or doing further searches. Computers are wonderful servants, but they take everything — including our mistakes — literally. If the results look strange to you, review your search commands carefully. The difference between

('Bob' AND 'Carol') OR ('Ted' AND NOT 'Alice'

and

'Bob' AND ('Carol' OR 'Ted') AND NOT 'Alice'

may turn out to be significant. A good way to check the search results is to make certain that a randomly-chosen record within the output actually does satisfy your search request.

From Ideal to Real: Some Inexpensive TRP Programs
Up to this point, we have been dealing with a hypothetical TRP. None of the programs that I discuss below does exactly what our hypothetical model does. Rather, each emphasizes one or more features described above. None of these programs costs more than $50, and most cost significantly less; some are available for the asking.

For each program, I give a brief summary and then a

description and evaluation of how the program works. These descriptions are summarized in Table 1. I also include information on how to obtain each program.

Originally, I intended to begin this section with a speed comparison across the programs, and to this end I tested each program using the ASCII transcript from a three hour unstructured interview — approximately 22,500 words. The slowest program I tested was a version of GREP, which took 70 seconds to go through the file; the other programs all had times of 30 seconds or less. Consequently, I have not included a speed comparison. The differences here are negligible.

Rather than focus on speed in deciding which program might meet your needs, I suggest that you consider the features that particular programs emphasize that might make them most useful in your particular work. One important factor to note is that, while the hypothetical TRP described above is controlled through command lines, many TRPs are menu-driven: You select the actions you want from a list, and the computer does the rest. These may be simpler to use for those unfamiliar with computers, or in classroom situations.

Golden Retriever

Golden Retriever, version 4.0, shareware — $39.95. Golden Retriever is a powerful TRP; it can be menu or command driven, and it has the capability to search for multiple-word phrases. It even has an adjustable fuzziness level. That is, you can make close guesses at the spelling of terms you don't quite recall, and Golden Retriever will often find them. The degree of "fuzziness" Golden Retriever will allow in a search comes preset to a reasonable level, but you can make the search more or less rigorous through a menu choice. Golden Retriever can make use of logical operators, as described above, but only in a very limited fashion. If you AND words together, for example, they will only match *exactly* the same pattern in the file — 'Bob' AND 'Carol' will only match *Bob Carol*; it will not match *Carol Bob* or *Bob Alice Carol*. The words in the file must not only appear in the same order as in the search criterion, but they must also be adjacent.

Golden Retriever's menus are clear and easy to understand. When Golden Retriever finds a word in a file, it takes you to the appropriate record and highlights the word on the screen; you may then use the cursor keys to choose how much of the surrounding material, if any, to save into an output file. One unusual feature allows Golden Retriever to run in the background, while you work in your word processor or other text entry program. Pressing a special key shifts you into and out of the Golden Retriever program, allowing you to search for data while you are working on a report, for example.

There is a preview version of Golden Retriever available, the Golden Retriever Pup. Golden Retriever Pup works exactly the same way as does Golden Retriever except that it will not read data files on a hard disk, which limits its usefulness considerably.

The Pup version is available via modem from computer bulletin boards, or for $10 from the National Collegiate Software Clearinghouse (NCSC), Duke University Press, 6697 College Station, Durham, NC, 27708. The full version can be ordered for $39.95 from Wesware, 42 Epping Street, Lowell, MA, 01852.

GREP

There are dozens of versions of GREP available, most of them in the public domain, posted on computerized systems across the country. If you have access to a modem, this is one way to locate a GREP program. If you don't, find a colleague or computer center person who can help you. Most microcomputer GREPs will explain themselves to you if you enter **GREP** or **GREP ?**. GREP is a good place to start looking at TRP systems because it is simple and cheap; you should be able to obtain a copy for free. Most (but not all) versions of GREP can save output to a file by appending the command '>', followed by a file name, to the end of the search request. Hence,

GREP 'bob' intrvw.txt >save.bob

saves the results of the search to the ASCII file SAVE.BOB.

Resnoter

ResNoter, version 1.0, (c) NCSC — $35. Resnoter is one of the most technically sophisticated programs I evaluated. It is the only one of the TRP programs reviewed here that uses indexing. This means that using ResNoter is keyword-intensive; if you want to use this TRP, you *must* insert extensive keywording in your data; ResNoter will not search raw text. From the keywords that you supply, ResNoter constructs a list of code words and their locations in the database.[10] If you ask for 'bob' AND 'carol', ResNoter need only look at the locations in the 'bob' list and compare them to those in the 'carol' list. It can then jump directly to the records that satisfy the request.

Because of this indexing, all logical operators are available and ResNoter is very fast. However, you must remember that if you add a new keyword you will not be able to use it until you have re-indexed the database. Indexing does not take long, but it is a step you must not forget in working with ResNoter. Another drawback is that ResNoter is loaded with menus. Menus should make life easier for the user, but ResNoter's menus are posi-

tively frightening because their operation is highly inconsistent. Still, if you need rapid access to large amounts of data, and if you are willing to insert keywords, ResNoter is worth learning. You can save output to a file, and you can choose what portion of the record to save (keywords, raw data, or both). ResNoter is one of a series of text retrieval and analysis programs published through and available from the NCSC.

Search

Search, version 1.3, public domain — free/$25. Search is one of the more complete TRP programs reviewed. It uses all three logical operators and, unlike Golden Retriever, is not limited to 'exact' logical matches. That is, Search scans the whole record to see if the logical requirements are matched, not just adjacent sets of words. 'Bob' AND 'Carol' will match not only *Bob Carol* but also *Carol Bob* and *Bob Alice Carol*. Search supports multiple levels of parentheses and can search for any combination of up to fourteen words and/or phrases.

One particularly nice feature, useful with logical OR searches, allows Search to note, either on the screen or in the output file, which of the logical search terms it matched in a given record. You can also have Search ask you whether or not to save a given retrieved record to its output file. An option allows Search to work like GREP, if you want to look only at line-sized records.

Search has two relatively minor drawbacks. First, it is mainly command-driven. To use it, you must learn to type in a sequence of commands. For example, if you gave this command:

SEARCH INTRVW.001 B =bob&carol > OUTPUT.TXT

Search would look for paragraph records containing both 'bob' AND 'carol' and saves the results to OUTPUT.TXT. These commands are not hard to learn, but may intimidate a first time user. Search provides a second, more limited search mode for beginners, which allows for only AND and OR operators. In this secondary mode, the TRP asks the user for search terms and filenames. Still, it is not as friendly as a menu-driven system like Golden Retriever.

A second drawback is that matched words are not highlighted in retrieved records when they are printed to the screen — if you are dealing with large records, this can make it difficult to find the exact point at which the match occurred.

Search is available free on computer bulletin boards or directly from its author for $25, which includes a subscription to future versions. Note however that, if you obtain SEARCH from a bulletin board, no donation is expected. For further information, contact Eric Bohlman, 1921 Highland Avenue, Wilmette, IL, 60091.

Table 1

PROGRAM NAME	VERSION	BOOLEAN LOGIC	FUZZY SEARCH	SAVE OUTPUT	MAX SIZE PER RECORD	PRICE
Golden Retriever	4.0	yes(1)	yes(2)	yes	no limit	$39.95 (3)
GREP	various	no	no	yes	1 line	free
Resnoter	1.0	yes	no	yes	no limit	$35.00
Search	1.3	yes	no	yes	no limit	free/$25.00
SeekEasy	5.0	no	yes (4)	no	2 lines	-------$30.00

(1) Golden Retriever uses Boolean logic to match only adjacent words.
(2) Golden Retriever allows the user to adjust the level of 'fuzziness.'
(3) A "sample" version is available through computer BBSs.
(4) SeekEasy's fuzzy search is not adjustabl

SeekEasy
SeekEasy, version 5.0, shareware — $30.00. SeekEasy is a slightly speedier but much less successful implementation of "fuzzy" searching than Golden Retriever, with considerably less flexibility. You cannot adjust the "fuzziness" level, and the program is limited to two lines of context around each word or phrase it finds. There is no way to save the results of the search to a file. In its favor, SeekEasy is *extremely* easy to use; type what you are looking for and, if it is in the file, SeekEasy will find it. Unfortunately, since it will retrieve the 100 closest matches (in no particular order), it will find a great deal of material you don't want, and you may have to search through all of that to locate what you asked the program to find for you in the first place. This program would be more useful for keeping an address list than for data searching.

SeekEasy is available from computer bulletin boards or for $10 from the National Collegiate Software Clearinghouse. The author requests a $30 contribution if you make use of the software, and that also entitles you to updated editions, when they are released. For more information, contact Correlation Systems, 81 Rockinghorse Road, Rancho Palos Verdes, CA, 90274.

Other Programs
In the course of this section I have limited myself to a discussion of public domain and shareware programs, with the exception of ResNoter. All of these programs are available for less than $50, and some can be had for free.

Potential users should be aware, however, that a large number of commercial programs exists designed for similar purposes. These include Ask Sam, Gofer, ZyIndex, Notebook II+, FYI3000, and the word processing package Nota Bene, which includes an interface to the FYI3000 text database system. Some conventional DBMS packages, such as DBASE IV, have added features that allow them to cope with large bodies of textual data as well. Potential users should also be aware, however, that the price of these programs can range from the moderate to the stratospheric. While I would not discourage anyone from investigating some of these programs, I have not found that the increased costs purchase significant increases in power or sophistication in TRPs.[11] What the increased costs *do* buy is support. If you are uncomfortable with, or inexperienced in the use of computers, it may be worth spending some extra money to gain access to software support personnel. For those who have moderate computer experience, however, public domain software and shareware come very close to being the proverbial free lunch, and I would encourage you to investigate those sources first.

Summing Up

Quantitative researchers, with their relatively simple data, have been the first to benefit from the computer revolution. But the increasing speed and power available through microcomputers makes even the complex textual data of qualitative researchers more accessible. This article has described the ways in which common, inexpensive, TRP systems may be useful in dealing with large quantities of interview data. The approaches described here can be applied as well to other textual data — field notes, for example, or archival research entered through text scanners. Any textual data can be made more useful through the application of simple computerized tools.

The availability of these tools does not, however, absolve the analyst of his or her responsibility. TRPs can only retrieve and display data — they cannot understand what those data mean, and they will willingly supply answers to queries whether those queries are motivated by theoretical understanding or conceptual blindness. Computers are always increasing in power, but never in intelligence, and it is worth remembering the first principle of data processing — GIGO [12] — whenever one sits down at a keyboard. Always think of the computer as an exacting but unimaginative research assistant, and you will not go far wrong.

Finally, you should be aware that TRP programs are rapidly increasing in power and flexibility and that, by the time you read this, there will probably be new versions available of most of the programs discussed here and a host of new TRPs as yet undreamed of. To find out about the latest programs, contact your computer center or local users' groups. A little time spent looking at the available TRP programs will be rewarded with a simple but powerful data retrieval tool.

* For helpful comments on earlier drafts of this paper, I would like to thank Theresa Marchant-Shapiro, Charles Tidmarch, Martha Huggins, Renata Tesch, and several referees, who shall remain nameless.

[1] Presented at the IASSIST 90 Conference held in Poughkeepsie, N.Y. May 30 - June 2, 1990.

[2] GREP stands for *general regular expression print*. Regular expressions are ways of expressing complex patterns of letters and numbers. GREP was originally designed to search through lists using these expressions and print the results on a teletype terminal.

[3] Public domain software is a body of programs placed by their authors into free public circulation: the programs can be freely copied, used and given away but cannot be sold for profit. Computer hobbyists often trade these programs and they are also available through electronic bulletin boards and services such as Compus-

erve. Finally, there are companies that sell public domain software through catalogs for a 'copying fee,' which is usually no more than a few dollars per disk. Public domain software should be differentiated from shareware, where the author of the software freely distributes his or her programs, but asks for a contribution from those who use them. Both public domain and shareware are excellent sources for useful and unusual programs. Note however that, for your own peace of mind, you should carefully test such programs. *Never* test new programs on the machine you use for storing interview transcripts and book chapters: Programmers sometimes, though rarely, accidentally release programs with bugs in them, and it's best to find out without destroying irreplaceable materials.

[4] GREP is an extremely flexible tool, capable of rapidly seeking out particular patterns in your text database. Rather than go into details here, however, I refer you to the support personnel at your institution. If you have access to a UNIX-based computer, however, you should be able to get a comprehensive overview of GREP with the following command:

man grep

This will display the pages of the UNIX manual dealing with GREP on your terminal. These will give you some sense of the power of the program.If you are using an MS-DOS based computer, on the other hand, your MS-DOS manual will give you an outline of how to make use of the FIND program.

[5] For computational purposes, and for the purposes of this paper, a paragraph includes all single-spaced text that occurs between sets of double carriage returns.

[6] All of the quotations in this article are constructs based on a series of unstructured interviews I conducted in 1988-89.

[7] Typically, you have an ASCII file if, when you use the MS-DOS "type" command to show your file on the screen, lines end without wrapping around from the right to the left, and you see only alphabetic, numeric, and punctuation characters on the screen.

Unfortunately, most of the more powerful word processors do not create pure ASCII files. WordStar and WordPerfect, to name two popular word processing programs, include special codes in their files to make printing easier. Such codes must be stripped out if the file is to be searched by most TRPs. Some word processing programs solve the problem of these codes with a built-in option to save files in ASCII format. In WordPerfect, for example, you should save your transcription into a "DOS TEXT" file. This will be an ASCII version of the file, with all special characters removed. For many other word processors, you will need a special conversion program. For the most part, such programs are available free or at a nominal charge. If your word processing program is incapable of writing an ASCII file, go to your college or university microcomputer lab or computer center, and explain what you need to do. They should be able to help you find a suitable conversion program.

[8] Fuzzy searching is a term that covers a great deal of ground. In general it means one of two things. If they do not find an exact match, some programs will look for words or phrases that contain many of the same characters in the same order as the search phrase. Others will seek words or phrases that are phonetically similar to the search phrase.

[9] We might think of each 'record' as being made up of a chain of dummy variables (words). Each word in the record indicates the presence of a characteristic, and the absence of a word indicates the absence of that characteristic. In any given search, therefore, we are trying to discover whether particular dummy variables are present or absent. At the same time, we will be ignoring most of the variables in a particular record — all of the words that do not appear in a search command.

[10] Given that you can only search for words that you have explicitly coded, you may want to think carefully about the amount of work involved in such coding before choosing this TRP. Its power comes in large part from a great deal of time and preparation on the part of the user. Coding 80 pages of interview (the outcome of the three hour interview I used to test programs) is, to say the least, a nontrivial investment of time.

[11] While some commercial program may have slight speed advantages over their public domain competitors, the major constraint on search speed is likely to be the access speed of the hard disk in your computer. Since this affects all programs equally, and is the major constraint on data retrieval, retrieval speed should not be given undue weight in deciding between two TRPs.

[12] "Garbage in, Garbage out."