ON ALPHABETIC SEARCHING IN VIDEOTEX SYSTEMS

H.A. Maurer
W. Rauch
I. Sebestyen

August 1981
WP-81-111

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
2361 Laxenburg, Austria

ABSTRACT

Of the four major types of interactive videotex systems currently being tested (Telidon, Teletel, Captains and Prestel-like) only one (Teletel) permits the use of alphabetic keywords for searching. It is contended that alphabetic keyword searching should be incorporated into future videotex systems. Methods of alphabetic keyword searching in the absence of alphanumeric keyboards are then discussed. A novel technique which has recently been implemented on Prestel-like systems is proposed as an interim solution until genuine alphabetic searching becomes available.

CONTENTS

ON ALPHABETIC SEARCHING IN VIDEOTEX SYSTEMS

H.A. Maurer, W. Rauch and I. Sebestyen

1. INTRODUCTION

This note is concerned only with a minor aspect of interactive videotex systems, hereafter called VTX systems. For a broader perspective of VTX the reader is referred to the literature (Woolfe 1980).

Alphanumerical Searching (Keyboard) versus Numerical Searching (Keypad)

The basic idea for VTX systems originated in the United Kingdom and is now known under its trademark Prestel. It involves the "upgrading" of color TV equipment with additional electronics, thereby converting it into a simplified computer terminal, which is hooked up to a VTX center via a dial-up phone line. To keep modifications of existing TV sets minimal, it appeared reasonable to use the standard remote control unit of the TV set as an input keypad. However, the use of such a keypad, having a fairly limited number of symbols, has serious limitations: basically, only digits and a few special characters are available for input. For this reason, Prestel was originally designed as a simple information retrieval system (see below); the huge potential for making genuine interactive use of the VTX-center computer and of already-existing knowledge and experience in information retrieval systems was ignored.

Following Britain's lead, other countries started to develop VTX systems. The Canadians made use of improved graphic facilities (Telidon): The Japanese took a facsimile-type approach (Captains), necessary because of the complexity of Japanese characters: The French (with Teletel and their Electronic Phone Directory) were the only ones to take a substantially different approach as far as keypad and searching techniques are concerned: they introduced alphanumeric keypads and full alphabetic searching while other major systems were still being designed with purely numeric keypads and, consequently, rather rudimentary searching techniques.

It is our contention that an approach involving alphanumeric keypads such as the French one is most promising and should be pursued in the future. This is not only because keyword searching as such is important (as will be demonstrated in Section 2, it can be handled fairly efficiently even with purely numeric keypads!), but because without alphabetic input, VTX must remain a simple retrieval-only system. With alphanumeric keypads it can turn into a versatile omnipresent, multipurpose retrieval and transaction system: properly equipped VTX terminals will become easy-to-use, inexpensive (through mass production) computer terminals, for the general consumer (business and residential) market, to be used for a broad variety of tasks (Maurer 1981) such as: information retrieval; transactions (bookings, reservations, fund transfers); electronic mail; entertainment; education; and personal and telecomputing.

To appreciate the difference between what Prestel or Prestel-like systems (e.g., as they are used in Great Britain, Switzerland, and Austria) can do and what VTX systems as outlined above could do, a brief description of how Prestel[†] works is necessary.

Information is stored on so-called pages or frames. Each page is identified by an integer. Each frame can point to up to 11 further pages, to be reached by typing either #, or one of the digits 0 to 9 (see Table 1). A page with number n can be accessed by either typing *n# or, if page m is currently displayed and m points via x to page n (where x is #, or 0,1,2,...,9), by typing the symbol x.

When a user dials up the VTX center he obtains automatically page 0 of the system. This page (like most others) offers a number of choices to the user, a so-called "menu." By selecting the appropriate alternative and typing the corresponding symbol, a further page can be obtained, etc.

For the discussion in Section 2 it is important to understand two points: *first*, typing *nm# (where n and m are positive integers) does not (necessarily) result in the same page as typing *n#m. (E.g., input *12# gives page number 12, but *1#2 can give a page with arbitrary number m, if page 1 is made to point to page m using digit 2). *Second*, no page can point to more than 11 further pages, but if page m permits choice k, that choice can be keyed in before page m has completely appeared on the

[†] From the point of view of information retrieval, Prestel, Telidon and Captains are quite similar.

Table 1. Uses of Prestel-Type Keypad

*0#	Return to start
*n#	Jump to page n
*#	Go back one page
*00	Repeat same page
**	Correct keying error
d	($0 \leq \alpha \leq 9$) jump to the page pointed at by pointer α
#	Follow-up frame, if existing

screen; in this case, the full page will never become visible. (Thus, suppose page m leads via four pages obtained by consecutive choices d_1, d_2, d_3, d_4 to page n; then typing d_1, d_2, d_3,d_4 in rapid succession will immediately give page n: the intermediate pages will not be shown and no time will be consumed in building them up).

The significance of the above rather technical points will not become apparent until Section 2. However, from what has been explained it should be clear that the search for information based only on the menu approach and numerical input will sometimes be rather cumbersome. The possibility to type in a keyword x if information on an item x is desired is certainly an attractive feature (currently only available in Teletel). In addition, an alphanumeric keyboard is essential for using VTX systems for sending messages, for ordering from a catalogue offered on VTX, for carrying out a reasonable dialog in a teaching application of VTX, for being able to use VTX as terminal for simple programming tasks, and so forth. Not to equip VTX terminals with alphanumeric keyboards tremendously reduces the value of VTX and permits only rudimentary services. The fact that in Britain, after two years of intensive campaigns by the British Post Office, Prestel has attracted only some 11,000 users is partly due, in our opinion, to the absence of alphanumeric keyboards and the possibilities they offer; and to the lack of real interactive capability in the British system.

In the past a number of arguments against the use of alphabetic keypads have been put forward. One such argument is the higher price for the somewhat more sophisticated keypads. Observing that most VTX users today are from commercial environments, the very small price difference (which only exists initially), seems insignificant when compared with the loss of applications otherwise incurred. Another frequent argument is that there would be an increase in complexity in using an alphanumeric board. We do not believe in this line of thought. Indeed, noting the circuitous ways used to try to overcome the lack of alphanumeric keys, our impression is quite the opposite: the lack of alphanumeric keys makes using VTX more complicated, not easier. However, the design of the keyboard may have to be modified from the QERTY arrangement of letters found on usual typewriters to an arrangement easier for the naive (= nontyping) user to use, such as on the French terminals, where letters are arranged in alphabetical order. Yet another argument against alphanumeric keypads is the size and weight of the keypad. Observing that current remote control units of modern TV sets have up to 25 keys and noting that hand-held boards with 20 "genuine" keys (to be operated with the pointing figure of the right hand) and 3 "escape" keys (to be operated with fingers of the left hand while holding the keypad), allowing a total of 80 characters (i.e., small and capital letters, digits and 18 special symbols) do exist (e.g., Dynaflex), it is clear that this argument is not valid either.

Summarizing, alphabetic keypads are of crucial importance to VTX systems, both for searching and for other applications, and there are no good reasons why they should not be used instead of purely numeric keypads in the future. However, as long as only numeric keypads are available, a kind of "pseudo-alphabetic search," as explained in the next section, may alleviate to some extent the problem of having no letters available.

2. ALPHABETIC SEARCHING WITH NUMERIC KEYBOARDS

The Basic Idea

As has been explained in Section 1, searching for information based on a given keyword is an important asset for the user of a VTX-system. Despite the fact that many VTX-systems currently only offer numeric keypads, attempts are being made to alleviate the above mentioned problem by providing in all of them an "alphabetic index" using a menu-type search: the user interested in a certain keyword selects one of a number of choices depending on the first one or two letters of this keyword (e.g., Aa-Af ... 00, Ag-Ba ... 01, etc.), and repeats the process with the next few letters until the desired entry is located.

This process of "alphabetic searching by narrowing down" turns out to be a reasonably cumbersome process if the alphabetic index in question is of any size at all: the user has to refocus his attention a number of times between keypad and screen, and to wait each time for the screen to be filled (typically 5-10 seconds) and to enter the appropriate choice.

Since Prestel-like systems and Telidon do not support any searching beyond the menu technique, it is generally agreed that evidently this is the only way to perform an alphabetic search.

It is our contention, and the main point of this note, that even with current systems and purely numeric keypads a more clever way of organizing an alphabetic search is possible. The basic idea is to associate a group of letters with each of the digits 1,2,...,9. By typing in a string of letters, a string of digits is obtained (in fact, creating a hash-code for each string of letters) which--by organizing the data appropriately--leads to the desired information.

More specifically, we associate the letters A,B,C to the digit 1, the letters D,E,F to the digit 2, ..., the letters YZ and the symbol "."(period) to the digit 9 (most conveniently by attaching little stickers with the appropriate letters on or below the numeric keys on the keypad) (see Table 2). In this fashion, we associate with each string of symbols w composed of letters and the period-symbol, a sequence of digits that we denote by d(w).

Suppose that the set of keywords used for a certain database is W and that each key w ∈ W has a number of frames of information associated with it, the first of which we denote by f(w). We choose a page with number n as start-page of our "pseudo-alphabetic index" (as we will call it henceforth). I.e., keying in *n# results in the display of that start-page. The main idea is to assign the frame f(w) to the page whose number is obtained by typing *n#w. (However, because of the way we have associated letters and digits, this of course amounts to inputting *n#d(w).)

It should be clear that searching using such a pseudo-alphabetic index is exactly the same as searching an ordinary alphabetic index--as long as "everything works out." Evidently, there are a number of problems that may arise and would have to be taken care of. As we demonstrate below this is easily done in each case, and does not lead to serious obstacles to using the proposed technique.

Problems

Problem 1 (Different keywords with same hash- code):

Clearly, it is possible that different words w and w' (of same length) yield the same hash code, d(w) = d(w'). This problem is easily solved by using the page accessed by *n#w (= *n#w') not to store the frame f(w) or f(w'), but to store a frame offering two choices leading to either f(w) or f(w').

Table 2. Proposed Assignment of Letters to Digits on Videotex Keypads

1	←-	A,B,C
2	←-	D,E,F
3	←-	G,H,I
4	←-	J,K,L
5	←-	M,N,O
6	←-	P,Q,R
7	←-	S,T,U
8	←-	V,W,X
9	←-	Y,Z,.

Problem 2 (Input of a word which is not a valid keyword):

If the user types *n#z, where z is a word not in W, one of two things may occur:

a. If a word $w \in W$ with $d(w) = d(z)$ exists the user is lead to information for the keyword w. To avoid difficulties with respect to this, each first frame f(w) associated with a word w should contain w in a clearly visible fashion. In this way, a user who keys in *n#z and obtains information concerning a word $w \neq z$ realizes that $z \notin W$.

b. If no word $w \in W$ with $d(w) = d(z)$ exists, inputting *n#z would ordinarily give a systems message "page non existent." If desired, this can be replaced by a special frame with a more elaborate message such as: "no keyword with the code d(x) exists. To return to start-page of index press 0, to ...." To access such a special frame F the pointers must be organized as follows: let z = uav, where u is the longest prefix of z such that d(u) is the prefix of some d(z), $z \in W$, and where a is a single letter. By definition of u, d(ua) does not occur as prefix of any d(z), $z \in W$. Hence the choice d(a) on the frame *n#d(u) can be used to point to F.

Problem 3 (Unnecessarily long input):

Suppose $z = ua_1a_2 \cdots a_m \in W$ (where a_i are individual letters for $1 \leq i \leq m$, $m \geq 1$) is a word such that u is as short as possible and d(u) does not occur as prefix of any d(z), $z \in W$. Thus, when the user has typed in *n#u, it is already clear that he intends to input *n#z and should be saved the trouble of typing the remaining m symbols. Hence, it is reasonable to associate the frame f(z) already with *n#u rather than with *n#z. Without further adjustments, however, this is not a good solution: the user who types in rapidly $ua_1 \cdots a_i$ ($i \geq 1$) will get a message "page non-existent" and will get the wrong impression that $z \notin W$. To avoid this dilemma and yet to permit access by short prefixes it is sufficient to ensure that the page *n#u points to itself with each of the digits, 1,2,...,9(!).

Problem 4 (The prefix problem):

This problem only arises due because in Prestel-like systems only one digit is processed at a given time. Thus, if two keywords w and z (but z longer than w) have the property that d(w) = d(w'), where z = w'z', then slowly keying in *n#z yields, on the way to the desired information, the page *n#w representing f(w). This is quite apt to confuse the user. To avoid such confusion it is probably preferable to insert an extra frame F as the page *n#w which offers the choice to either reach f(w) or else to continue to complete the desired keyword by typing in further letters.

Problem 5 (Long prefixes):

A number of different keywords $w_1,w_2, \cdots w_t$ may have the property that $d(w_1)$, $d(w_2)$, ..., $d(w_t)$ have a long common prefix u. To shorten the input process for the user, one may choose the shortest v, such that v is prefix of $d(w_1)$, $d(w_2)$, ..., $d(w_t)$ (but of no other keyword $w \in W$) and one could insert an extra frame F as the page *n#v, which allows t choices leading to $f(w_1)$, $f(w_2)$, ..., $f(w_t)$, respectively. To prevent the user from going beyond the page *n#v, that page should point to itself for every digit 1, 2, ..., 9.

Probability of Collision

In above fashion, a purely numeric keypad and a system designed only for numeric menu choices can be used for alphabetic searching by typing in the keywords almost as if a full alphabetic search facility were available. Of the five problems mentioned above, the last three (and problem 2 to some extent) arise in any alphabetic keyword system. Only problem 1 arises merely because of the hashing technique proposed. Although "collisions" are easily resolved as explained, the user might well find it annoying that (rather than getting the desired information directly) he is forced to make one additional choice at the end. Hence it is important to have some feeling for how often a "collision" (different keywords w, w' with same code d(w) = d(w')) will occur.

This is a classical problem from the theory of hashing and data-structures; see e.g., Maurer (1974). A uniformly distributing hashing function will give roughly α collision per key, where α is the loading factor, and will give α collisions in total. Assuming that we consider only words with a length of five letters (for a rough estimate), we have an address space of 9^5, i.e., roughly 60,000. E.g., a list of 240 keywords this gives a loading factor $\alpha = 240/60000 = 0.004$. The probability for a collision is thus less than 1/2 percent.

This is still a fairly pessimistic estimate since only short words were considered: the address space for actual English words is somewhat higher, resulting in a still smaller probability of collision.

The first experiment with such a pseudo-alphabetic index was carried out in the Austrian videotex pilot-trial, see e.g., Maurer (1981): information on 260 Styrian[†] towns and villages was prepared in this fashion. No single collision occurred, in good accordance with above calculations. Experimental results with untrained personnel showed that the time required to find a specific keyword with the pseudo-alphabetic index is about half of that required when using the narrowing-down approach.

A final remark concerning the above mentioned Austrian experiment may be of interest. As start-page of the pseudo-alphabetic index the number 35228 was chosen, since d(INDEX) = 35228. Thus, in the Austrian videotex trial, typing *INDEX#z, where z is the name of any Styrian town or village will give information on that location.

[†] Styria is one of the nine provinces of Austria.

Conclusion

The pseudo-alphabetic index is a possible "crutch" (to be used only as long as necessary), but it is not a substitute for a full alphabetic keypad and genuine alphabetic searching, which, as outlined above, seems to be essential. Not only is it fairly hard to manually structure the data correctly (hence requiring software for that purpose, see Aurenhammer (1981)), but also real alphabetic input (e.g., for messages) is not allowed, despite the fact that the index can be used in this fashion in a somewhat cumbersome way: to type a letter, one first lists the corresponding key (i.e., 1 for A,B,C; 2 for D,E,F; etc.) and then 1, 2, or 3 depending on whether the desired letter is the first, second, or third on the key just used. Thus, A would be encoded as 11, B as 12, C as 13, D as 21, and so on; the end of words would be indicated by typing a zero.

As clumsy as the method may sound, if keys are equipped with the appropriate lettering, both encoding and decoding is directly possible (without any memorizing or pencil and paper). The method is being used for sending messages in the Austrian VTX pilot trial (e.g., allowing people to register for certain events) and is used in the field trial in the Federal Republic of Germany in a still somewhat clumsier version for sending "number letters" to, say, Axel Springer Pub. Co. (Hoefele 1981).

3. INDEXING VERSUS SEARCH-TREES

Even if this method of "numerical coded alphabet" turns out to be of limited practical value (as we actually hope, if the general trend should be towards alphanumeric keyboards), it has some conceptual aspects for VTX, since it bridges the gap between the two main information-access schemes, namely search-trees and indexing.

In case of search trees, all accessible elements are organized into sets and subsets of increasing specificity and are ordered into a "tree-structure." An example is Dewey's Decimal Classification, which covers the whole universe using such principles. Less sophisticated systems are familiar to us from most organizational schemes. The VTX access trees are one of the most recent applications. A search tree is simple to understand, simple to construct, and is almost self-explanatory in use. It is suitable for specific applications as well as for global attempts.

However, there are basic limitations to the system of search trees: the decisions taken in building up the tree are irreversible and restricted to only one dimension; the structure of the tree is rigid and cannot be adjusted to changes in the system's environment; browsing through the system is almost impossible; and in complex cases, a high number of branches has to be passed through before ending up at the "leaves" of the trees.

Therefore information retrieval systems now use a different approach for accessing information: indexing. In this method, brief descriptions of data are organized in a file. The elements (= data) may be indexed with a varying number of descriptors according to different aspects. The index-terms may be chosen "freely" or from a controlled vocabulary (= thesaurus). For retrieval purposes they can be arranged according to the rules of Boole's logic. Using computer technology even very large index files (at present up to ten billion words(Burns 1981)) can be organized as inverted databases; information science invented very "sharp" retrieval instruments operating on such inverted index files (including natural language access).

The "numeric coded alphabet" method for VTX-systems combines elements of search-trees with elements of indexing. The mode of access strictly follows a decimal tree, and is thus fully suitable for VTX-logic. But this tree-structure follows the digital construction of the alphabet, not the structure of the reference material. So, although at first glance this method shows characteristic elements of a tree-structured information-retrieval system, it definitely is of an index type, since it operates on inverted files.

With the combination of a tree-structured access path and an inverted index file, the described method opens up the possibility of an index-oriented retrieval approach for user of the VTX system, even without alphanumeric keyboards. To make all other advantages of index-oriented information retrieval systems available for VTX systems, it will be necessary to introduce appropriate software for the information supplier function of the system. This could be realized by using VTX as gateways for external computer capacity already equipped with such possibilities. In that way, VTX can be developed from simple information distribution networks into sophisticated information retrieval systems.

72322202252210

REFERENCES

Aurenhammer, F. 1981. Bildschirmtext Alternativ Index. Diplomarbeit. Graz, Austria: Institut fuer Informationsverarbeitung, Technical University Graz.

Burns, C. 1981. Information Storage and Display. Journal of the American Society for Information Science 32(2):145.

Maurer, H.A. 1974. Datenstrukturen und Programmierverfahren. Stuttgart: Teubner.

Maurer, H.A. 1981. Bildschirmtextaehnliche Systeme; Studie fuer das BMfFu W.

Hoefele, J. 1981. Bildschirmtext--der kurze schnelle Weg zum Leser. Bildschirmtext Seminar at Schloss Laxenburg, Laxenburg, Austria, June 17, 1981.

Woolfe, R. 1980. Videotex. London: Heyden.