Timothy C. Craven
University of Western Ontario, London, Ont.
School of Library and Information Science

Simulation of Visually Displayed Indexes*

Craven, T. C.: **Simulation of visually** displayed indexes.
In: Intern. Classificat. 7 (1980) No. 1, p. 21–24

Potential applications of simulated visually displayed indexes include training of indexers and testing, evaluation, and selection of indexing systems; control of index simulator parameters could be valuable in experimental observation of index use; index simulators might aid on-line users in selecting browsing display formats. Simulation has certain advantages over the use of real indexes for these purposes.

A pilot NEPHIS index simulator, written in PET BASIC, generates "pseudostrings" (hypothetical subject descriptions) from a user-supplied input string and permutes and sorts the pseudostrings to produce a simulated index, or "pseudoindex". More sophisticated systems for the extrapolation of subject descriptions might be developed in future. (Author)

1. Introduction

The term "visually displayed indexes" is meant to include both traditional printed indexes and less traditional forms, such as index displays produced on demand as aids in online retrieval and the relatively less volatile indexes on COMfiche. Nor does it exclude nontextual indexes, such as colour indexes, and indexes making use of various forms of graphic display.

Computer simulation has received relatively little attention in the field of indexing, and still less attention has been paid to the possibility of a computer system to simulate a visually displayed index, though Heaps (1978) has suggested the use of mathematical models in the simulation of bibliographic databases for the training of online searchers.

This article is concerned in general terms with the computer simulation of visually displayed indexes, and more specifically with a pilot system for using a small amount of real indexing data in producing mockups of a particular kind of index on a microcomputer. Of what use might such a simulated index be? Perhaps the most obvious area of application is in teaching good indexing using a specific indexing system. In order to see whether a given index entry is good, the student must be able to consider the type of context in which it is likely to appear in the final product. Will the position of the index entry in relation to other index entries, similar and dissimilar, be helpful to the user, either in retrieving specific items or in browsing? The simulated index can provide hypothetical contexts for the student's efforts at index entry production.

Another possible use of simulated indexes lies in the testing, evaluation, and selection of existing or proposed indexing systems. Different algorithms for the shunting of elements in subject descriptions to produce index entries (Lynch 1973; Austin 1974; Neelameghan 1975; Cohen 1976; Craven 1977; Farradane 1977; Craven 1978; Anderson 1979) need to be tried out in a variety of situations. Many display formats are possible, especially for highly detailed indexes; these formats need to be compared. Subtle invalidities in input to computer-assisted indexing systems may be clarified by the generation of hypothetical entries; the fruits of such clarification might be improved instructions to indexers, or more sophisticated error detection routines, or a redesigned indexing system.

The possibility for the user to choose among a number of options by specifying values for various parameters is likely to be a desirable design feature for an index simulator. By selectively manipulating these values, a researcher could control such variables as similarity among subjects indexed and familiarity of terminology. Indexes with qualities precisely defined in this way could provide valuable "environmental" control in experimental observation of such aspects of index use as time required to locate a known item.

Visually displayed indexes of course include index pages or "screens" displayed by online systems in response to browsing commands. In the online context, the simulated index can be seen as a possible selection tool for users who wish to specify their own browsing requirements.

How does simulation compare with possible alternatives, such as the use of "all-real" printed indexes, whether full-size or compiled from a small sample of indexing data? An advantage of simulation over the use of full-size indexes is that it obviates the costs associated with compiling, storing, and accessing large quantities of actual indexing data. An advantage over the use of sample indexes is the greater ease with which it can bring out features and problems of indexing systems, especially of indexing systems designed for highly detailed subjects or very large files.

A further advantage lies in the possibility of tailoring a simulated index to meet specific needs. For example, it may be desired to highlight certain features of a particular indexing system (such as subarrangement under main headings) without cluttering the view of the observer with special cases and illustrations of other, possibly distracting, features. Where useful, a simulated index may be made as "ideal" or as "realistic" as necessary.

Having in mind the potential significance of simulated visually displayed indexes for information decision making, I undertook a pilot project of writing a simple printed index simulator. The language used was BASIC: the equipment was a PET 2001-8 microcomputer (Commodore 1978); the indexing system was NEPHIS (Craven 1977).

2. NEPHIS

NEPHIS is termed a "string index language" by Svenonius (1977, pages 338–344), because the indexer supplies the computer with "strings" of index terms and additional tags, each string representing a subject. The computer uses each string to produce a set of permuted index

entries, each of which is a complete representation of the original subject. When sorted and formatted by the computer, the index entries combine to form a printed or otherwise visually displayed index of high browsability. I developed NEPHIS with four principal objectives in mind:

1. it should be easy for the indexer;
2. the computer algorithm should be easy to implement using any language capable of character-string manipulation;
3. running the resulting program should be economical; and
4. the index produced should satisfy the users.

Usually a NEPHIS string takes the basic form of a noun phrase with other noun phrases "nested" within it. The indexer needs to know only four tagging characters ("⟨", "⟩", "?", and "@") and the four commands that these define. The symbols "⟨" and "⟩" are used to set off a nested phrase; the symbol "?" is used to indicate a connective (an element that will appear according to specific rules in some permutations but not in others); the symbol "@" is used to suppress an entry under the phrase it precedes. For example, a typical input string produced by a NEPHIS indexer is:

@EVALUATION? OF ⟨OXIDES? OF ⟨IRON⟩⟩? FOR ⟨@MANUFACTURE? OF ⟨FERRITE⟩⟩* M38--8/53

This instructs the NEPHIS program to produce the index entries

OXIDES OF IRON. EVALUATION FOR MANUFACTURE OF FERRITE * M 38–8/53
IRON. OXIDES. EVALUATION FOR MANUFACTURE OF FERRITE * M 38–8/53
FERRITE. MANUFACTURE. EVALUATION OF OXIDES OF IRON * M 38–8/53

3. The pilot NEPHIS index simulator

The user of the pilot simulator first types in a NEPHIS input string. (Fig. 1 illustrates this initial step.) The simulator begins by analyzing the user's input into its component parts and storing the results of this analysis in the microcomputer's memory. The simulator then manipulates the components in various ways in order to produce a set of simulated input strings, or "pseudostrings", which are variants of the original input string. The original string and the pseudostrings are used to generate sets of permuted index entries and "pseudoentries", which when sorted, form the simulated index, or "pseudoindex". On a conceptual level, the production of pseudostrings may be viewed as an extrapolation or projection of likely subject descriptions from a given subject description.

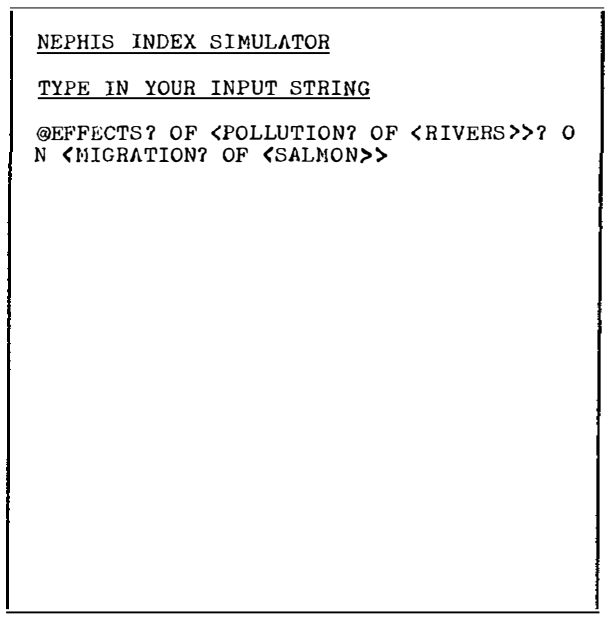

Fig. 1: Screen display. Typing in of NEPHIS input string

What kinds of pseudostrings a simulator produces is in part a function of the structure of the indexing system, and in part dependent on the purpose to which the pseudoindex is to be put. The pilot simulator makes use of three basic methods for generating pseudostrings, all three of which are for the most part facilitated by the structure of NEPHIS.

Method 1 is the deletion of one or more nested phrases. The resulting pseudostrings are usually *generic* in meaning to the source string. From the input string illustrated in Fig. 1, this method would yield, among others,

@EFFECTS? OF ⟨POLLUTION⟩ ON ⟨MIGRATION⟩

which in turn would lead to the pseudoentries

POLLUTION. EFFECTS ON MIGRATION

and

MIGRATION. EFFECTS OF POLLUTION

Compare these pseudoentries with the entries generated from the original input string:

POLLUTION OF RIVERS. EFFECTS ON MIGRATION OF SALMON – RIVERS. POLLUTION. EFFECTS ON MIGRATION OF SALMON – MIGRATION OF SALMON. EFFECTS OF POLLUTION OF RIVERS – SALMON. MIGRATION. EFFECTS OF POLLUTION OF RIVERS

On the conceptual level, we can see the inference from the existence of a subject "effects of pollution of rivers on migration of salmon" that there is likely to be a subject "effects of pollution on migration".

In Method 2, the pseudostring is a phrase nested at some level within the string from which it is derived. One pseudostring produced by this method from the input string illustrated would be

RIVERS

The pseudostrings generated normally are not generic in meaning to the source string, but will represent *broader* concepts (for the definition of which, see Soergel 1974, page 78): a document about the effects of pollution of rivers on the migration of salmon is necessarily also *about* rivers, even though the effects dealt with in the document are in no way a *kind* of river. By way of comparison, not only is a document about effects of river pollution on migration of salmon about effects of pollution on migration; the effects discussed are also a subclass of effects of pollution on migration.

Method 3 involves the substitution of different element for parts of the original string. The descriptions created are usually of *related* (or apparently related) subjects. The illustrated input string, for example, might yield

@EFFECTS? OF ⟨AABANKS⟩? ON ⟨MIGRATION? OF ⟨SALMON⟩⟩

– the nonsense element "AABANKS" being substituted for the phrase "POLLUTION? OF ⟨RIVERS⟩". Nonsense substitutes were adopted in the pilot simulator because

they are flexible and occupy relatively little space in the program. As discussed below, more context-sensitive substitution could be employed in more sophisticated simulators. Austin (1975) has noted one value of nonsense examples in teaching a system for the production of permuted indexes: students are not distracted by the suspicion that the computer really understands what they mean. Nonsense elements may also be picked specially (as in the pilot system) to illustrate filing order.

Two other possible methods, not incorporated in the pilot system, should also be mentioned. One, Method 4, would consist of the insertion of extra nested phrases into the source string. A typical result would be

@EFFECTS? OF ⟨POLLUTION? OF ⟨RIVERS? IN ⟨ZYZZYLIA⟩⟩? ON ⟨⟨MIGRATION? OF ⟨SALMON⟩⟩

With certain well defined exceptions, pseudostrings generated by insertion would represent concepts that are *hyponyms* (Hutchins 1975, page 43) of the concept represented by their source string; that is, the source string would be related generically to the pseudostrings derived (the reverse of what usually occurs with Method 1, deletion).

Method 5 would entail embedding the source string in a suitable longer phrase, as in

ATTITUDES? OF ⟨AABANKERS⟩? TO ⟨@EFFECTS? OF ⟨POLLUTION? OF ⟨RIVERS⟩⟩? ON ⟨MIGRATION? OF ⟨SALMON⟩⟩

This method would generally produce a pseudostring representing an apparent *narrower* concept, and may be seen as the reverse of Method 2.

Figure 2 summarizes the workings of the five methods of pseudostring generation and the conceptual relationships corresponding to these methods.

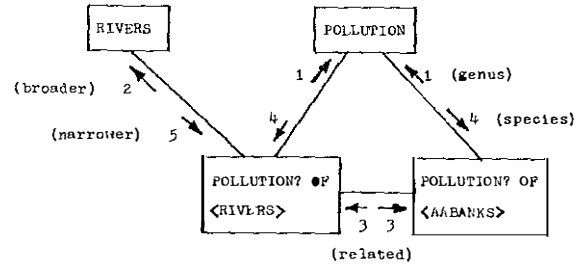

Fig. 2: Pseudostring generation methods and corresponding conceptual relationships

It can be seen how greater or lesser use of the various methods will tend to simulate different kinds of index. The first two methods lead to pseudoentries that are quite similar to the original subjects, but less specific. Methods 3, 4, and 5, if nonsense terms are employed, could be used to emphasize the results of unfamiliar terminology. Method 4 might suggest to an indexer the possibilities of greater depth of indexing; Method 5, those of a shift in emphasis.

The first page of the pseudoindex is displayed automatically (Fig. 2). Pseudoindex entries are in "hanging indentation", a fairly simple format to program. The user may request the next page (if any) by typing a "Y", or return immediately to the string input step by typing an "N".

Typical time between completion of string input and display of the first page of the pseudoindex might be as

```
PSEUDOINDEX          PAGE 1
AABANKS
AABANKS. EFFECTS
AABANKS. EFFECTS OF POLLUTION OF
       RIVERS
AABANKS. EFFECTS ON MIGRATION
AABANKS. EFFECTS ON MIGRATION OF
       SALMON
AABANKS. MIGRATION. EFFECTS OF
       POLLUTION OF RIVERS
AABANKS. POLLUTION. EFFECTS ON
       MIGRATION OF SALMON
AABANKS OF POLLUTION OF RIVERS
AABANKS OF RIVERS
AABANKS OF SALMON
AABANKS ON MIGRATION OF SALMON
MIGRATION. EFFECTS OF AABANKS
MIGRATION. EFFECTS OF POLLUTION OF
       RIVERS
MIGRATION. EFFECTS OF ZYZZYLS
MIGRATION OF AABANKS. EFFECTS OF
       POLLUTION OF RIVERS

NEXT PAGE?
```

Fig. 3: Screen display. First page of pseudoindex

short as one second – suitably quick for practical use in indexer training or the like – provided the program were rewritten in machine language. This estimate is based on a processing time of around five minutes in the current interpretive version and a report of an improvement in speed of about three hundred times when translating a program from PET BASIC to machine code (Covitz 1978).

Because of the high rate of word repetition in the pseudoindex, it is possible to pack a large number of pseudoentries into a small amount of memory by storing the pseudoentries not as actual text, but as strings of pointers to the text elements that comprise them. Thus the apparent size of the pseudoindex can be far greater than the capacity of the machine being used. Moreover, the occurrence of highly similar entries in the pseudoindex simulates aspects of still larger indexes. In short, simulation enables an inexpensive microcomputer to stand in for a much larger system for certain purposes.

Chief areas of application of the pilot system are seen in instruction and demonstration, as well as in development of more sophisticated simulators. Observations of the semantic relations between pseudostrings and input string and of the limitations on procedures for pseudostring generation may also have implications, in need of further exploration, for the development of future permuted indexing systems.

4. Future possibilities

Two methods of extending the capabilities of the pilot simulator were mentioned above. Various directions seem to be open for increasing further the sophistication of index simulation in future systems. Two of these directions are marked by use of statistical models for estimation of index entry probabilities and by use of semantic information to widen the scope of inference.

In the first direction, Heaps (1978) has already suggested the employment of mathematical models of

descriptor distribution over documents in producing more lifelike mockups of information retrieval systems. It is a short step from here to the use of models of descriptor distribution over subject statements to increase the verisimilitude of simulated permuted indexes. For example, different frequency distributions for different terms could provide more realistic cutoff rules in pseudostring generation than is possible in the pilot system (where the total number of pseudostrings is arbitrarily limited to 31, regardless of constituents).

In the second direction, we might begin on a fairly primitive semantic level, where, say, presence of the connective "BY" might be used to indicate that a word belonging to the class of agents could be substituted for the following nested phrase. A machine-readable thesaurus might be accessed as an aid in inference-making of various degrees of complexity. At a simple level, such a thesaurus could, for example, tell the simulator that "FISH" could reasonably be substituted for "SALMON" in our sample string.

The possibilities for subject inference in changing the focus of interest within a network of closely linked concepts should also not be overlooked. For instance, from the subject "migration of salmon" a sophisticated index simulator should be able to extrapolate the subject "migrating salmon". In linguistic terms, such an extrapolation might be seen as a transformation of surface structure while retaining the same deep structure. I am currently engaged in work relating to closely related subjects of this kind in a slightly different connection.

* Revised and expanded version of Craven, Timothy C. "Microcomputer simulation of large permuted indexes", Proc. ASIS Annual Meeting 16 (1979) p. 168–173.

References

(1) Anderson, J. D.: Contextual indexing and faceted classification for databases in the humanities. In: Proceedings of the ASIS Annual Meeting 16 (1979) p. 194–202

(2) Austin, D.: PRECIS: a manual of concept analysis and subject indexing. London: Council of the British National Bibliography 1974.

(3) The University of Western Ontario, School of Library and Information Science, Course 780SS, Summer, 1975. "Exercise 1".

(4) Cohen, S. M., Dayton, D. L., Ricardo, S.: Experimental algorithmic generation of articulated index entries from natural language phrases at Chemical Abstracts Service. In: J. of Chem. Inform. and Computer Sci. 16 (1976) No. 2, p. 93–99

(5) PET 2001–8 personal computer user manual. 1st ed. Palo Alto, Ca.: Commodore Business Machines 1978

(6) Covitz, F. H.: Life for your PET. In: The Transactor 5 (1978), (Sept. 30) p. 22–31

(7) Craven. T. C.: NEPHIS: a nested phrase indexing system. In: J. of the Amer. Soc. for Inform. Sci., 28 (1977) No. 2, p. 107–114

(8) Craven, T. C.: Linked Phrase Indexing In: Inform. Proc. and Management 14 (1978) No. 6, p. 469–476

(9) Farradane, J., Gulutzan, P.: A test of Relational Indexing integrity by conversion to a permuted alphabetical index, In: Intern. Classificat. 4 (1977) No. 1, p. 20–25

(10) Heaps, H. S.: Computer simulation of document data bases (Paper presented at the Conference of the Canadian Classification Research Group, Melrose, Ontario, May 5–7, 1978).

(11) Hutchins, W. J.: Languages of indexing and classification: a linguistic study of structures and functions. Stevenage, Herts.: Peter Peregrinus 1975.

(12) Lynch, M. F.: Petrie, J. H.: A program suite for the production of articulated subject indexes. In: Computer J. 16 (1973) No. 1, p. 46–51.

(13) Neelameghan, A.; Gopinath, M. A.: Postulate-based Permuted Subject Indexing (POPSI). In: Library Sci. with a Slant to Doc. 12 (1975) No. 3, p. 75–87

(14) Soergel D.: Indexing languages and thesauri: construction and maintenance. New York: Melville, 1974.

(15) Svenonius, E., Schmierer, H. F.: Current issues in the subject control of information. In: Libr. Quarterly, 47 (1977) No. 3, p. 326–346