SECTION D : Computing

This section is intended simply to constitute a short note on our computational procedures, to indicate our approach and resources, and not in any way to provide an exhaustive technical description. It summarises first our treatment of data, and second our program facilities.

I Data

It will be evident from previous sections in the Report that our collection, and especially document, data is much simpler than that used in operational systems in that we concentrate only on index terms and can disregard bibiliographic detail, etc. It will also be clear that the static character of test collections, even where natural language rather than controlled language indexing is involved, means that bulky verbal material can be replaced by simple numerical data, with identifying number names for keywords or stems. A further point is that as the request and relevance sets are known in advance, files can be set up embodying all term matching information, for example, which may be more economical for repeated search experiments than regular files. Our standard data formats are therefore primarily for a range of simple data structures with numbers (typically simply positive or negative integers including zero) as primitive elements. We have found that while particular bodies of data having these structures may have very different interpretations, our processing requirements for many different purposes can be effectively met by common programs based on standard formats.

Of course, such ideas are by now common places of data management, but in the past information retrieval projects have tended not to aim at generality of data structures of processing, largely because they have been relatively short term, or have been closely associated with operational services having their own specific formats and operations. We have noticed an increasing payoff, in research over a relatively long period, from our emphasis on standardisation, and in particular have profited from our habit of setting any collection up in standard form before conducting any experiments with it, even though this may be a non-negligible effort. Our standard form collections are also, we hope, more portable and easily exploited than some of our collections have been in the forms in which we initially received them.

1 Data Formats

From the interpretive point of view our data structures may be divided into two main groups, and this is a convenient way of presenting them here; but it must be emphasised that different data sets with the same format may have quite different interpretations, and that there may, as will be described below, be overlaps between standard form collections in predominantly different formats.

a b data

Our initial data sets were mostly in a b form. A data set in this form consists of a list of a elements (numbers) each having a list of b elements. The list of a's is terminated by our standard terminator '/', and each b list is similarly terminated. Examples of a b data are:

a set of documents (names) each characterised by a list of terms (names)
a set of requests (names) each characterised by a list of relevant documents (names)
a set of terms each characterised by a list of documents.

The constraints on a b data are that the a's must be in ascending order, without duplicates, and equally the b's in a specific b list. There is, however, no requirement that either a's or b's in a list should be successive, and a or b lists may be empty (but correctly terminated).

p q data

A natural associate of a b data is the p q list or information list: this consists of a single list of number pairs terminated by '/': i.e. each p number has an accompanying q. Examples of p q data are:

a set of documents each with the number of terms it contains
a set of requests each with the number of its relevant documents
a set of terms each with the number of documents in which it occrs.

Clearly in this structure each p mentioned must have its q, even if this is only zero. The p's must again be in ascending order without duplicates, but need not be successive.

a b c data

One extension of a b data is a b c data, where each b has an accompanying c element, i.e. each a has a list of b c pairs. Examples are:

a set of documents each characterised by a list of terms, each term having a weight
a set of terms each characterised by a list of cooccurring terms and cooccurrence values.

The general constraints on a b c structures are as for a b structures, with the additional requirement that each b has its c.

a b b data

As noted in the text, the UKCIS profile data called for treatment rather different from that adopted for the other collections, as the profile terms consisting of word fragments, word strings etc. could not be treated in a simple way as items in a document-derived term vocabulary list: a word fragment corresponds to a set of document words defined essentially arbitrarily at search time and referring to only one of many possible groups. This led us to data sets acceptable in an experimental context with a closed collection, embodying information about the occurrence of each profile term in the document set, i.e. representing request document matching. The format is a list of a's, each a having a list of b's (referred to as b1's), with each b1 having a furtherlist of b's referred to as b2's; each list is terminated in the usual way. The format is in fact quite generally useful, though originally designed for a specific purpose. Examples are

a set of profiles each with a list of profile terms, the latter each with a list of documents
a set of profiles each with a list of relevant documents, the latter each with a list of terms.

The constraints are similar to those for a b data, namely that the a's, b1's and b2's must all be in sort order; an a may have no b1 list, and a b1 no b2 list.

The corresponding information list has an already defined format, namely a b c. Thus for the first example above we may have a document count for each term.

Once these schemes are familiar, a variety of extensions are easily made and handled, for example a b c c data; the general concepts are also naturally applied to cases where one type of element is a word: for example we may have an analogue of a b data consisting of document names followed by

word lists with regular terminators, or of a p q list consisting of term names each accompanied by a word. The use of common formats has in particular made us appreciate that programs originally thought of in one context for one specific application may have a far wider utility. For example, a program designed to supplement request relevant document lists with the latter's term lists is equally a program to expand request term lists with the latter's document lists. Of course, not everything is as tidy as the foregoing would suggest; but the general picture is solid enough: for example the initial processing of data obtained from elsewhere will typically produce many intermediate ad hoc files; and some program outputs giving, for instance, retrieval perofrmance or data analyses, fall outside the framework.

2 Standard Collections

The standard formats for our collections consist of a set of stream embodying the basic document request and relevance information. We have two types essentially associated with non-matched and matched request-document sets respectively. The former, relying primarily on the a b format, has been used for most of our collections, the latter, relying on the a b b format, has mainly been used for the UKCIS profile collections, though as it has proved convenient for some experiments we have begun to extend some of our older collections in this way.

The two forms are as follows:

(a) non-matched collection

This represents a collection as available for retrieval experiments, with associated information.

Part 1 Stream	0	Documents with term numbers
	1	As 0, but documents serially numbered
	2	Requests (serially numbered) with term numbers
	3	Requests with relevant document numbers
	4	Conversion list for real-serial document numbers
	5	Term dictionary - i.e. words representing the alphabetically first member of each keyword group based on a common stem, with numbers, numerical order
	6	Term dictionary, alphabetical order, if different from 5
	7	Documents with terms
	8	Requests with terms
	9	Inverted documents - i.e. inversion of 0
	10	Inverted requests - i.e. inversion of 2
	11	Inverted relevant documents - i.e. inversion of 3
	12	Document term frequencies - i.e. summary of 0, giving the number of terms per document
	13	Request term frequencies - i.e. summary of 2
	14	Relevant document frequencies - i.e. summary of 3
	15	Distribution data : see below
Part 2 Stream	0	Term document frequencies - i.e. summary of 9
	1	Term request frequencies - i.e. summary of 10
	2	Relevant document request frequencies - i.e. summary of 11
	3	Conversion list for original-serial request numbers.*

Distribution data

This analyses P1 S12,13 and 14 and P2 S0, 1 and 2.
Using the Keen Documents Distribution information as an example, we have

* Owing to extreme antiquity there are small variations in the standard pattern for some collections e.g. for stream 4 the order may be real-serial, or serial-real. There is also some variation in whether the names of empty items are given e.g. in stream 9

MIN = identifying number of the shortest item (or first equal shortest) and its length: thus document 317 has 1 term
MAX = identifying number of the longest item (or first equal longest) and its length: thus document 512 has 20 terms
NOS = number of items: thus there are 797 documents
TOT = total of item lengths - in this case total postings: thus there are 5729 postings
AV = average item length: thus there are seven terms per document. NB that for some old data the integer part only, not rounded, is given.

There follow the numbers of items having given lengths: thus there is 1 document of length 1, 30 of length 2, 48 of length 3,etc.

(b) matched collection

This represents a collection after document matching for request terms (which typically consists of a fragment, but may also consist of a multi-word string, which may match different words in different documents, or different strings), with associated information.

Part 1 Stream	0	Requests (serially numbered) with document numbers each with matching request term numbers, the latter internal to requests and referred to as idnos
	1	As 0, but document numbers serial
	2	Nil
	3	Requests with relevant document numbers
	4	Coversion list for real-serial document numbers
	5	Term, i.e. word dictionary, usually without stop words
	6	Term dictionary, alphabetical order, if different from 5
	7	Documents with terms, i.e. words, as texts, including stop words
	8	Requests (serially numbered) with idnos and terms, and any other e.g. Boolean structural information
	9	Inverted matched requests, i.e. inversion of document number/idno information of 0, per request
	10	Nil
	11	Inverted relevant documents, i.e. inversion of 3
	12	Requests with document term frequencies, i.e. summary of 0
	13	Request term frequencies, i.e. summary of part of 8
	14	Relevant document frequencies, i.e. summary of 3
	15	Distribution data: see below
Part 2 Stream	0	Requests with term document frequencies, i.e. summary of 9
	1	Nil
	2	Relevant document request frequencies, i.e. summary of 11.
	3	Conversion list for original-serial request numbers

Distribution data

This analyses P1 S12,13 and 14 and P2 S0 and 2

The format is the same as for the standard collection, but where three rather than two level data is involved i.e., P1 S12 and P2 S0, the identifying numbers of the shortest and longest items involve a specific request number reference.

Note that as the collection embodies matching information there is no direct distribution information about words and documents, but only about matching terms and documents, where terms are specific to requests.

II Programs

By now we have built up a substantial general purpose suite of programs. The programs fall (roughly) into seven classes, respectively concerned with

(a) creating standard data
(b) manipulating standard data
(c) information gathering
(d) classification
(e) weighting
(f) retrieval
(g) evaluation

Our philosophy has been that it is more useful, given the unexpected directions that research can take, to devise solid data formats and to allow many individual programs for carrying out relatively specific operations on a body of data. In early projects we built a few large, multi-purpose (i.e. multi-option) packages, but these proved costly and inflexible, and we have found our current approach much more satisfactory. Our programs are written almost entirely in BCPL, and are run on the University's 370/165 computer, on which an extremely good service is provided. Runs typically take, say, 1-5 seconds for simple processing of smaller data sets, such as inverting a request set or carrying out coordination term matching on the 42 x 200 C200T collection, and 20-30 seconds for data set generation or retrieval for the U27000P collection. Our suite contains over 200 different programs, amounting to perhaps 20,000 lines of code.

Creating standard data

As our collection data has been obtained from various sources programs have been required to extract material of interest from a range of miscellaneous tape and file formats, and from a mass of mixed information. These have unfortunately not always been trivial, but are not usually of general interest. The more important, general programs in this group are those for word processing: they include those for removing stop words, for sorting word lists, for setting up dictionaries, for suffix stripping, and for dictionary matching. The last, for example, is done by two-letter table lookup for simple word matching, and by character inclusion check followed by string matching for front-truncated fragment matching.

Later stages in collection creation are dealt with by standard data manipulation routines.

Manipulation standard data

This group covers a range of programs for operating on data in the standard formats described earlier. In particular there are corresponding programs for a b data and for a b b data, and some of the programs are designed to deal with data sets too large for in-core operations. Altogether there are a large number of programs in this group, including ones for sorting, inverting, intersecting and merging data sets in different formats, and for a wide range of selection and expansion procedures using one data set, say of p q type, to control derivation from another of a b or a b b type. A characteristic example is a routine to replace the b elements in one a b data set by those in a second.

Information gathering

The routines in this group are chiefly relatively simply frequency counting ones, for different standard formats, but it also includes programs for comparing data sets, for collecting the distributional information given

in the standard collection characterisations, and for providing more detailed information like that embodied in the Q values for individual terms described in Section B. The cluster hypothesis test program also falls into this category.

Classification

Earlier research involved a considerable number of classification programs but these were not transferred to the 370, and the current range is fairly limited. It comprises programs for obtaining cooccurrence arrays, for computing similarities, and for constructing simple star-type classes.

Weighting

The routines in this group cover the provision of the various forms of weighting we have studied for both a b format and a b b format collections. Thus there are simple programs for obtaining collection frequency weights and rather more complex ones for computing the four types of relevance weight in both their straightforward and special case versions.

Retrieval

Corresponding to the two types of data are two main sets of retrieval programs producing output by matching levels. They include programs for unweighted term matching and for different kinds of weighting: we have found separate programs for the various cases more efficient than one global one with many options. The programs are mainly quite simple, but those for relevance formulae F3 and F4 are more complex since term absence as well as presence is covered. For a b data the programs make a single pass of the document file against the inverted requests and relevance judgements; for a b b data the basic data already gives request document matching information so the single pass is only against relevance, and in some cases weighting information. There is a separate set of programs, necessarily more expensive, for producing ranked output, and a further set for Boolean matching. In addition, programs are available providing detailed performance for individual requests.

Evaluation

Though the main retrieval programs produce our standard recall and precision results, we have other routines for characterising performance according to our various alternative methods, for example giving cumulative effectiveness.

Overall, our program sets include routines suited to most contexts, allowing us to conduct equivalent experiments with data in different formats. New collection preparation requires a good deal of time, particularly if the collection is large, but many of our data processing operations and comparative retrieval experiments can be performed in a very straightforward way. We can also by now carry out new tests within our general range of tests quite easily, with little additional programming effort.

Note on stemming

The stemming procedure is fully described in Andrews 1971. It is of a straightforward kind involving the use of a single suffix dictionary and contextual rules for identifying genuine suffixes and for determining stems. Thus '-ual' is not removed from 'equal' and 'control-' is found as stem in 'controlling'. The program works fast and generally correctly. It should be emphasised that what stems are found, i.e. what words are conflated, depends on the input word list, since there is no stem dictionary.