5 Access to catalogue files: indexing

5.1 Introduction and guide to the chapter

In the sense in which it is used here, the term "index" refers to a file or set of files which is used to provide a means of access to the bibliographic records in the catalogue file.

It is not always misleading to think of an index in this sense as being rather like a back-of-the-book index. In many reference retrieval systems an index does indeed largely consist of an ordered list of words or phrases each associated with an ordered list of record numbers. However, the analogy should not be relied on: some IR systems (the Cambridge University OPAC is an example) do not have an *ordered* index, but it is not true to say that such a system does not have an index.

The chapter falls fairly naturally into two sections: readers who are interested in indexes from the users' viewpoint — that is, in the contents and use of indexes — should read Sections 5.2 to 5.5; the remainder of the chapter gives a brief introduction to some of the computational aspects of index storage and searching, as much as is relevant to an understanding of the design of Okapi.

Most of the sections in this chapter cover their topic from a general point of view, followed by an account of how it was applied in the design of Okapi. They cover the *content* of indexes (by source and by type) (Section 5.2), the *display* of indexes and the closely connected question of *filing order* (Section 5.3), then techniques for *deriving and selecting index keys from catalogue files* (Sections 5.4 and 5.5). These are followed by the sections on the more *computational aspects*: Section 5.6 gives a general discussion of storage of and retrieval from indexes and Section 5.7 gives a fairly detailed description of the methods used in Okapi.

5.2 OPAC index contents

However indexes are stored and structured, they contain the *keys* by which bibliographic records are retrieved. In Section 2.4 OPACs were categorised by their mode of indexing. OPACs derived from circulation systems are indexed on field contents — that is, index keys consist of the contents of a MARC field or subfield, possibly truncated on the right. Sometimes (for example in Geac circulation systems) these keys contain a portion from a second field, and are of fixed length. Indexes of the second category of OPACs — those based on or similar to online reference retrieval systems — consist in the main of individual *words* from the bibliographic records.

The first type of indexing is referred to as *phrase* or *pre-coordinate* indexing, and the second type as *keyword* or *post-coordinate* indexing (cf. Section 2.4).

There is also a third type of index key. Cataloguing systems used as OPACs (e.g. OCLC, SWALCAP) often also use "derived" or "acronym" keys: a simple example would be the first four characters of the personal author main entry (MARC 100 $a) followed by the first four characters of the title (245 $a after removing the number of non-filing characters given by the second indicator, typically a definite or indefinite article). A key consisting of surname + initials could be considered as a derived key (since it is constructed from separate MARC subfields), although one which would appear natural to users. Some IR systems include word stems such as **medic** and **philosoph**. These also are derived keys.

5.2.1 Access points — the sources of index keys

The phrase indexed type of OPAC generally allows access by personal names and corporate names from author and sometimes name added entry fields, and by title, often including uniform, series and part titles. Some OPACs index shelf mark or class number, and most of the American systems index LCSH.

The keyword systems use words from all or most of these fields (a class number or shelf mark counting as a word). There is an obvious problem about what to do with personal names. Should a name be indexed as separate words — surname and forenames, by surname only or by surname with initials? There is a discussion of this in Section 5.4.1.

In the systems which use acronym keys, these are commonly derived from author and title fields, and sometimes from other sources such as subject headings.

All types of OPAC may include in their indexes such keys as control numbers, OCLC numbers, copy or accession numbers, which are rarely of use to the public but are used by staff.

5.2.2 Data types

In catalogue records there are at least five types of indexable data:

personal names recognisable as such (there are of course also personal names in title fields which are not identifiable by machine — see Section 4.3.4)

corporate names recognisable as such

natural text: title-like data

controlled subject headings assigned by a cataloguer

codes and numbers — shelfmarks, class numbers, local numbers, control numbers, dates of publication

The data type influences the way keys are extracted or constructed, the way they are stored, and whether and how they should be displayed.

Each of these data types is discussed in more detail in Section 5.4.

5.2.3 Okapi index contents

The Okapi index contains items extracted from all the MARC tags listed in Appendix 2 except control numbers, accession numbers and other local data, and publication and edition data.

These are:

personal authors and added names

words and phrases from corporate and conference authors and added entries

words and phrases from all title-like fields

words and phrases from subject headings and verbal feature headings

personal names from name subject headings

date of publication

Dewey number(s)

Keys are truncated when necessary, currently phrases at 40 characters and other keys at 31 characters.

There are also three types of derived key:

surnames from all identifiable personal name keys

a 4/4 title/author key for the main title and every (corporate or personal) author and added name

Dewey numbers truncated at segmentation points given by MARC 082 $b

At early stages in the indexing process much information about the source and nature of keys is stored, but in the final index only the following ten types of key are distinguishable:

AU author and added entry personal names (surname + initials) and corporate author and added entry phrases

AW author and added entry surnames and corporate author and added entry words

TPH title, series title, part title phrases

TW title, series title, part title words

CL Dewey numbers

DT dates of publication

T/A 4/4 title/author keys from the first TPH and all AU

GS subject heading phrases, words, surnames and surnames + initials from subject headings, title words, corporate name words (overlaps with AW and TW)

GP all personal names and surnames (overlaps with AU, AW, GS)

GCL Dewey numbers and truncated Dewey numbers (contains CL)

The key types or roles are referred to as *beasts* in Okapi jargon. In the Okapi prototype, the specific item search uses the first four of these beasts and the T/A keys, and the subject search uses GS (Chapter 7).

5.3 Filing order and the display of indexes

5.3.1 Browsing and index order

In traditional IR systems — most of the older computerised and manual IR systems — indexes are for the user. Some OPACs allow users to see indexes, others do not.

Whether indexes are displayed or not, the ordering of their entries may be important, because this may determine the sequence of records in a browsing display. For example, index entries

economics of adv/chiplin
economics of adv/reekie w
economics of adv/schmalen

can be used to generate the display

Economics of advertising.	**Chiplin B**
The economics of advertising.	**Reekie W D**
The economics of advertising.	**Schmalensee R**

by selecting records from the source file in the order in which their keys are stored in the index.

In the case of *phrase* indexes, display of the indexes themselves can be useful, but whenever possible these should rather be used to give rise to displays containing more information, such as the brief title records in

the example above. In response to an author search for **Einstein** one OPAC gives a display similar to the following:

EINSTEIN

1. Albert (3 books)
2. Alfred (2 books)

Enter line number :

This is clear and useful, but since brief records for all the books by both the Einsteins could be shown on one screen, would it not be even better to offer the brief record display? (In fact the OPAC by which this example was suggested asks for a "choice" even when there is only *one* author satisfying the request.)

A browsing display of *subject headings*, with the number of items posted to each, is more often useful than an author or title index display: the display of alphabetically adjacent headings may help the user to find suitable controlled terms. Subject headings may be stored in a separate authority file rather than as an index derived directly from the bibliographic file (Section 5.5.2). Browsing is discussed more fully in Section 7.3.1 and also mentioned in Section 9.4.5.

Keyword indexes, in contrast to *phrase* indexes, should not be displayed — they are used invisibly by the system, and would be unhelpful to most users. An exception might be made for words from subject headings, but automatically produced keyword indexes are generally quite "dirty" and contain terms such as "adv", "dh", "econmics", "gt", which confuse users and do not encourage faith in the system. Some keyword systems even use post-coordination for personal names (**Henry James** from **Henry** AND **James**), and few would find a use for a list of assorted forenames and surnames.

5.3.2 The readability of index displays

If indexes are to be displayed they must be made as readable as possible. Personal names should look like personal names, with initial capitalisation retained. Numbers should, if possible, retain their punctuation. It is helpful to retain also capitalisation and punctuation in acronyms — "bbc" may be comprehensible but "it" for "IT" or "I.T." is not. Note that acronyms are like phrases or names rather than words (after all, they

represent phrases), and they should be included with these for display purposes.

5.3.3 Character coding and filing order in indexes

Assuming that an index is ordered, there are some rules that almost every user would expect to be obeyed — that letters and digits should file separately, that letters file in alphabetical order with upper and lower case being indistinguishable to the system, and that digits file from 0 to 9.

Character coding

Inside the computer characters are represented by codes (bit-patterns) which have an order imposed by the CPU of the computer. There are also conventional coding schemes, such as ASCII and EBCDIC, for translating characters as entered at a keyboard into internal codes. They all have the following properties: that digits file consecutively from **0** to **9** and that letters file consecutively from **a** to **z** and from **A** to **Z**. They do not agree on anything else. In most codings, upper and lower case letters file in separate sequences, so that, for example, **a** is less than **b** but greater than **Z**, and there is no agreement about the position in the collation sequence of blanks, punctuation and other non-alphanumeric characters. This is not so important as might be expected, because it is simple to alter the collation sequence to suit one's needs. A more serious problem is that of handling a character set large enough to cater for the needs of multilingual data. Most computer indexing systems do not attempt to deal with such tasks as distinguishing the French noun **thé** from the English definite article **the** while ensuring that these two words will file adjacently.

The effect of character coding on filing order

Leaving aside digits for the moment, and assuming that upper and lower case letters will not be distinguished in key comparisons, the designer of a computer indexing scheme has to decide how to code (1) blanks, (2) hyphens, (3) punctuation, (4) representations of letters which are not in the alphabet provided by whatever display device is to be used, and (5) characters which may occur in bibliographic records but do not fall into any of these categories. It is the treatment of blanks and punctuation

which has the most noticeable effect on the filing order of keys in phrase indexes.

Word-by-word or letter-by-letter order?

Word-by-word order is obtained by including blanks in the collation sequence, giving them a value lower than any character which can form part of a word. It is the form of ordering which traditional librarians may feel happier with.

If blanks and all punctuation are ignored, letter-by-letter order is the result. This causes **Thoms, Bertrand** to file before **Thom, Sylvia**. It is the scheme used in some forms of directory. A considerable number of OPACs appear to be using it. The advantage of letter-by-letter order is that it is easier and more economical to implement, both at the indexing, and, more importantly, at the searching stage. There is no need for elaborate parsing procedures to determine that **Salmon S R = Salmon, SR = Salmon S.R.** It also deals rather satisfactorily with some of the hyphenation problems, provided hyphens are treated as blanks. Its main disadvantage (apart from the fact that it may occasionally confuse people) is that some information is being thrown away in ignoring blanks: this can cause spurious identification of keys (**black bird = blackbird** and **Salmons R** is indistinguishable from **Salmon S R**), resulting in occasional false drops.

There seems to have been little work done on the relative merits of letter-by-letter and word-by-word orderings from the point of view of scanning speed and accuracy [1].

Hyphens

Letter-by-letter ordering deals with hyphens by ignoring them. In keyword systems, they should be dealt with according to their function. When used for attaching a prefix (e.g. **non**, **hyper**) they should be ignored, since there is often no agreed spelling for such compounds. When used to coordinate two or more meaningful words — the assumption here is that a word is meaningful if it is not a member of a list of prefixes — hyphens should be treated as blanks, and also the whole compound should enter the index. Using this rule, **user-friendly** gives **user**, **friendly** and **user-friendly** in the index.

PUNCTUATION

Punctuation can normally be ignored, as can apostrophes and quotation marks. Note that "—" and " - ", representing dashes, should be treated as blank.

DIACRITICS AND NON-DISPLAYABLE CHARACTERS

Some characters which are in the MARC exchange tape character set cannot be represented on many VDUs. Diacritics can be ignored and characters such as Polish L, Scandinavian O and Thorn filed and displayed as L, O and TH.

OTHER CHARACTERS

Other characters include ampersand (treat as AND), brackets, parentheses, dollar and pound signs (ignore). The solidus (/) may most conveniently be treated as a hyphen.

5.3.4 Filing order and display of the Okapi index

After certain substitutions have been made (such as **Mac** for **Mc** and the replacement or deletion of non-displayable characters) all types of key are ordered according to the same procedure.

The collation sequence for the character set is:

blank and hyphen (equal)
0 - 9
a - z and A - Z (upper and lower case indistinguishable)

All other characters are ignored (although many of them are retained). Repeated blanks are reduced to single blanks.

Since blank files before alphanumeric characters, word-by-word filing order is obtained, and because punctuation is ignored keys which differ only in punctuation or other non-filing characters will be treated as identical, and will become amalgamated during the indexing process.

Numbers will file in a peculiar way (although modern librarians may not be too upset — numbers often file in the same way in COM output). **100** will come before **20**, and decimal points are ignored. Luckily MARC records do not contain many numbers with decimal points, except Dewey

numbers, and these file properly because they all have the same number of digits to the left of the decimal point. Also, years file correctly because most of them have the same number of digits.

DISPLAYABILITY OF OKAPI INDEX

In the Okapi index, personal names and title phrases — AU and TPH (Section 5.2.3) — are displayable. Names retain capitalisation, hyphens and apostrophes, as do phrases. Phrases also retain most of their internal punctuation, brackets, parentheses and quotation marks, and an indication of truncation when they are over the maximum indexed length. In the prototype these are the only two classes of index terms which are displayed. In a more fully developed version Dewey numbers (full or truncated), and possibly surnames, would be displayed. It is felt to be a fault in the system that subject phrases are not distinguishable in the final index from subject words, partly because they could usefully be displayed.

The fact that each key is only stored once in the Okapi index (Section 5.7), regardless of its source(s), occasionally produces inconsistencies in display, particularly between words and phrases, because a key can be both a word and a phrase (for example *title* **Economics** and title and subject *word* **economics** — whether or not the index entry **economics** retains its initial capital depends on whether its first occurrence in the source file was as a phrase or a word). The same applies to surnames like **Wood**.

5.4 The construction of indexes for bibliographic records

Indexes may be constructed manually, automatically with manual intervention or completely by computer. We are not concerned here with the purely manual methods (although they are arguably the best for small specialised collections).

Rules for automatic and semi-automatic indexing are of two types: rules whose application depends solely on the source file and those which use data external to the source file — stop lists, go lists, authority files (Section 5.5).

The method of constructing index terms depends on the mode of indexing (phrase, word or derived key), the purpose(s) for which the index is to be used (subject searching, specific item searching, whether it

is to be displayed to the user), and on the type of data in the field being indexed.

It is worth pointing out that many of the same processes which are applied to a source file when producing an index must also be applied at search time to users' search statements. Some designers appear not to realise this: it is common to find systems where a title phrase search will fail if the user enters the title with a leading article. In general, procedures which cannot be applied during searching, such as those dependent on knowledge which will not be available during searching (for example knowledge of the language of a title) or those which depend on access to large files or tables, should not be used in indexing.

In the following subsections, there is a discussion of the first four of the data types referred to in Section 5.2.2, with examples, recommendations and comments. No detailed algorithms for the extraction of keys are given, but as an example there is an outline of a procedure for keyword indexing of title-like data.

5.4.1 Personal names

The simplest procedure is to concatenate the $a and $h subfields of MARC 100, 600 and 700 fields to make keys like **Austen Jane**. Many of the keyword systems make separate keys from $a and each forename in $h in personal name fields. This immediately illustrates one of the shortcomings of purely post-coordinate systems: a search for **Henry James** cannot fail to retrieve also **James Henry**. One system avoids most of these false drops by using a second derived key consisting of the first letter of the surname followed by the first letter of the forename, so a search for **Henry James** is submitted as **henry AND james AND jh** (which still retrieves books by **James Henry Jackson**). On the positive side, with such systems it doesn't matter whether the user enters a search for **Henry James** or for **James, Henry**. Also, they avoid difficulties which arise from users not knowing which is the entry element of a name (**Lloyd George**, **Mao Zedong**, **Van Rijsbergen**).

Generally though, particularly in quite large collections, some pre-coordination is highly desirable. On the other hand, consider a search for **Lawrence, DH** in an index containing *only* full keys ($a + $h) of the form **lawrence david herbert**. Depending on the design of the search functions, a user who enters the probably unambiguous personal name **Lawrence, D H** either has to do a good deal of browsing of index terms

or brief records (and filing order becomes quite important) or a quite elaborate "multiple truncation" has to be performed, the index being scanned from **Lawrence D.**. (**D H** having been recognised as initials). As always, one has to find a trade-off between precision, processing demands and storage.

If storage is not a serious constraint, then about as good a solution as can be achieved is obtained by storing for every name the three keys (sometimes the first and second will be identical) **Lawrence, David Herbert**, **Lawrence, D H** and **Lawrence**.

Stop lists are not used for personal names, but a system with an authority file can provide valuable cross references. If there is a MARC 900 tag (personal name cross reference) arising from the field being indexed, then this should also generate an index entry. The only OPAC we have seen which does this is the Cambridge University system.

In a keyword system hyphenated names should probably be treated like hyphenated words. Single character keys should not be omitted (**U Thant**). With any mode of indexing, readable display requires that hyphens and apostrophes be retained, but they should file as blank and nothing respectively. **Mc** or **M'** should file as or be replaced by **Mac**.

PERSONAL NAMES IN THE OKAPI INDEX

These come from field one (author main entry), field five (added names) and field nine (subject headings) of the Okapi source file. Personal names in these fields are identified as such in the source file (in field nine they are only identified where their origin was a MARC 600 field). No pre-processing is done except the replacement of leading **Mc** and **M'** by **Mac**. Any non-alphanumeric characters apart from apostrophes and hyphens would be removed, but there are unlikely to be any.

The "phrase" key consists of the entry element followed by an arbitrary number of initials separated by blanks (or hyphens) (**Taylor A J P**, **Sartre J-P**). This is how personal names are stored in the source file. The average number of initials is just under two. It would not save an appreciable amount of disc space to "pack" the initials (**Taylor AJP**) because the occurrence of names with double-barrelled entry elements and no initials means that **Tong Zhou** would be parsed as **Tong Z H O U** unless an "initials-indicator" were used.

The surname key is derived algorithmically from the phrase key, by taking that portion of the phrase which lies to the left of the first blank (or the entire key if it doesn't contain a blank). This doesn't work properly for names like **Lloyd George** (surname key **Lloyd**). A sensible algorithm would work by removing initials from the right of the phrase key until it came to an item longer than one character, but this is one of the pieces of code which was never written.

5.4.2 Corporate and conference names

These resemble both personal names and title-like phrases. Keyword indexing is attractive because it is often difficult to remember the exact wording — is it **Institute** or **Institution of Electrical Engineers**? The OPAC user's problems in accessing corporate names will be familiar to anyone who tries to look them up in telephone directories. However, the leading words in corporate names cluster heavily around words like **Great** (Britain), **Institute**, **Committee**, with the meaningful and memorable parts of the name often being towards the end. Some OPACs include items like **Great Britain** and **Committee** in the corporate name stop list so that **Great Britain. Department of Health and Social Security.** becomes **Department of Health and Social Security**.

If pre-coordinate indexing is used, cross references from acronyms and variant forms should be provided whenever possible.

In any case most library users do not recognise that a corporate body can be an author, so, however they are indexed, corporate name keys are not very much used. Among advanced searchers (those who chose "COMMAND" mode) of the University of California's MELVYL in May 1982, about 2% of searches were explicitly for corporate authors (and only about one-third of searches used COMMAND mode — in the simpler "LOOKUP" mode, an author search accesses both the personal and corporate name indexes) [2].

Corporate names often contain some indication of the subject content of an item, and this is very often in a MARC $c subfield (subordinate body, related body etc). The $a element, particularly when it is "name of government" can cause false drops in subject searches. When corporate names are used for subject indexing, a stop list should include very

common $a phrases (**Great Britain**). Words like **department, conference, committee** although of little or no use in subject searching are harmless enough.

In practice, the choice of indexing method for corporate names may be influenced by a decision as to whether or not they should be stored separately from personal name keys.

CORPORATE NAMES IN THE OKAPI INDEX

These occur in the same Okapi source fields as do personal names, from which they are distinguishable. The phrase and word keys are extracted exactly as the title keys (Section 5.4.3). The phrases are of limited use where they contain more than one MARC subfield (strictly they are not then *phrases* but several concatenated phrases): **Open University** is useful, but few searchers will enter (nor should they have to enter) **Great Britain. Board of Trade. Copyright Committee, 1951**. A future version of Okapi should index the subfields separately.

5.4.3 Titles

Here is an example of a typical term extraction procedure for title-like data in a post-coordinate system

1. Divide the field contents into words.
2. Replace non-displayable characters by their displayable equivalents (e.g. Polish L by L) and discard diacritics.
3. Convert capital letters to lower case.
4. Discard characters other than letters and digits and hyphens.
5. If a word doesn't contain a hyphen, do step 8.
6. (The word contains a hyphen.) Pass a copy to step 8, and separate a second copy of the result into words.
7. Delete "words" which were prefixes in the hyphenated phrase if they are in a list ("non", "in", "de", etc).
8. Remove items which are only one character long.
9. Remove words which appear in the title stop list.
10. Output the resulting terms.

Example: the title

Non-custodial and semi-custodial policies.

would give the following index terms

non-custodial custodial semi semi-custodial policies

The above term extraction rules are only intended as an example, and are oversimplified in several ways: they do not provide a means of distinguishing between "**IT**" and "**it**", nor do they make any attempt to handle input which is inconsistently or incorrectly punctuated (words separated by a comma but no blank, for example). Note that two or more full stops between single letters (**B.B.C.**) must not be treated as word separators. The punctuation of *numbers* expressed in digits also creates problems: in English, commas but not full stops may be removed, but the reverse applies in many other languages; the safest course may be to retain punctuation in numbers in an index, but to ignore it when searching (see also Section 5.3.4).

Titles are more easily and accurately remembered than are corporate names, and phrase indexing is rather effective for specific item searching (Section 2.4.1). That indexing *keywords only* is not ideal for titles is illustrated by the following nice example of a false drop in a title keyword search:

Sense and sensibility

retrieved a work with the subtitle

sense of impending crisis, refinement of sensibility, and life reborn in beauty

However, if titles are to be used to provide input to a subject index the keyword approach would normally be taken so both pre- and post-coordination are needed.

A compromise between phrase and keyword searching is offered by the modified Geac installation at the University of Sussex, which does not offer Boolean combination searching on keywords, but provides a degree of pre-coordination by generating a 16-character key starting at the beginning of each (unstopped) title word. For example the title

Britain's married women workers

gives rise to the keys

Britains married
married women wo
women workers
workers

of which the first can be used as a title phrase key, and all four can be used as subject keys. The fact that the first is useless (as a subject key) and the fourth too broad is less important than the highly pertinent nature of the second and third.[1]

Some points to be borne in mind when deciding how to index titles:

from MARC 245 and 440 only $a ("title proper") is needed for phrase keys — people don't usually recall subtitles; on the other hand, when present, $b ("other title information") often contains better subject content so subtitles must enter into keyword indexing.

statements of responsibility should not be indexed, as the names are not identifiable as such, and occur elsewhere in the record; it is unfortunate that with all its elaborate structure, MARC does not provide the means of searching for a specific person in a specific role (editor, illustrator etc) — see Section 9.4.5.

Titles in the Okapi index

Title-like entries occur in the second and fourth fields of the source file. The fourth field may contain more than one series and/or part title, but these are identified. Subtitles are not identifiable with certainty, but they are separated from main titles by **blank colon blank**.

A title is pre-processed to rationalise punctuation. This is mainly to try to compensate for typographical mistakes in records which were retrospectively converted: any comma, semicolon or colon not followed by a blank has a blank inserted after it, double hyphen intended as "dash" is replaced by **blank hyphen blank**, as are **hyphen blank** and **blank hyphen**. If there is any punctuation preceded by a blank the blank is removed (mainly this only affects " : " between title and subtitle). **Blank hyphen blank** is then replaced by a single blank, as are any repeated

(1) Complete rotation of titles, permitting a KWOC-like display, would of course be very expensive on disc storage. It has been suggested by Mischo [3] and by others as a means of generating printed subject-rich indexes from titles.

blanks. The phrase is then output, truncated to forty characters if necessary; if it is truncated, a dollar sign is appended.

The procedure for extracting words is applied to the (untruncated) phrase resulting from the phrase extraction procedure. The rules for word extraction are very simple, since the preprocessing has resulted in text in which every word is followed either by a blank or by an end of field marker. All non-filing characters are ignored, and a word ends when the next character is either a blank or an end of field character. Acronyms are preserved automatically, whether or not full stops separate the initials. Hyphens are not properly handled (again, this was something there has not been time to write); a simple rule of thumb is used: if a hyphen appears in the second, third or fourth character position it is deleted, otherwise it is retained. There is no special treatment of numbers.

Finally, the word is output unless it is one character long or in the stop list (Section 5.5.1).

5.4.4 Subject headings

It is difficult to generalise about the indexing of subject headings as their nature varies so much from one library to another. In general there are three types of item which one might extract from subject headings. First, there are long, LCSH-like headings such as

Islam—Africa, North—History

These are useless in an OPAC index *except* for browsing purposes. Even for browsing, their usefulness is severely limited unless they are rotated. Similar headings derived from PRECIS strings may be found in the MARC 083 Verbal Feature Heading tag.

Then there are personal name headings from 600 and short phrases from subfields of 650 and 651 (topical and geographical LCSH). These can usefully enter a subject index. (Often these subfields are single words.)

Finally, individual words can be extracted to allow purely post-coordinate searching.

Methods of term extraction do not differ noticeably from those used for other textual data. A minimal stop list may be used.

SUBJECT HEADINGS IN THE OKAPI INDEX

These are held in the ninth field of the Okapi source file, and consist of words, names or short phrases separated by full stops. Names are marked as such when they occur in the MARC 600 or 610/611 tags; identifiable names are indexed as described in Section 5.4.1. The phrases are indexed both as words and as phrases, the procedure used being identical to that used for titles (names in topical subject headings give rise to a separate key for each element, just as names in titles do).

5.5 Stop lists and authority files

In most IR systems some common terms are taken as *stop words* and are not indexed. A sensible system will ignore the same words if they occur in users' search requests. *Authority files* are lists of preferred or accepted forms of names or other headings.

5.5.1 Stop lists

What goes into a stop list depends on the degree of specialisation of the file and on the type of searches that are to be made. Subject searching requires a much larger stop list than is needed for known item keyword searching. If a file covers a narrow field then many words should be stopped which could not be stopped in an OPAC containing items in many subject areas.

In phrase indexing from MARC records, leading definite and indefinite articles are given by the second indicator in most of the title-like fields, so these can be stopped. This is not as helpful as it might be because the same procedure cannot *automatically* be applied to users' input (Section 5.4), and hence it is still necessary to have a list of words which are or may be articles. Some OPACs simply fail to find a title if the user enters it with a leading article; some at least issue a warning. One problem here is that it is often not possible to tell whether a word should be stopped unless the language is known. French and Italian "L' " are particularly troublesome, as apostrophes are almost universally ignored in indexing and must be removed from search statements.

It is arguable that no words except initial articles should be stopped in a keyword system used for specific item searching. There are distinct titles

Systems of organization and **Systems organization**, although in a small file such coincidences will be rare.

The well-known hyperbolic law of word frequencies means that there are a small number of very frequent words. One approach is simply to exclude from an index to be used for specific item searching the ten or twenty commonest words — these can best be determined by doing trial indexing runs, since the distribution of words from MARC records is different from the distribution of words in other sources of text (see Table 5.2).

A subject keyword index for a wide range of subjects needs a different approach. There is an illuminating discussion of the use and dangers of stop lists in subject retrieval systems by Bell and Jones in [4].

As well as articles and common conjunctions, most or all prepositions should be stopped. Care should be taken not to exclude words such as **system**, **general**, **approach**, **theory** which are generally "noisy" but meaningful in some contexts, either in specific subject areas or when adjacent to certain other words (**System analysis**, (army) **generals**, **theory of general relativity**, **approach roads**). An examination of a large number of subject statements submitted to an OPAC will show that there are many common words which rarely if ever occur. These include auxiliary verbs and many pronouns. One reason for large stop lists in traditional IR systems, particularly in-house systems, is that indexes are for browsing. In OPACs, word indexes should not need to be displayed, so this does not apply.

For all types of searching, a large stop list will increase recall but increase the number of false drops. Ideally, stopping procedures should be context-sensitive, so that, for example, **system** might be stopped unless adjacent to **analysis**, **theory** or **design**, or, in a system designed to produce output ranked in probable order of relevance, some approach may be made to probabilistic indexing: terms are assigned weights which depend not only on their relative frequency over the whole file but also on their role in a specific record. Thus, to take a simple example, words from a corporate name field might be assigned a lower "prior weight" in a subject search than words from a subject heading field.

Whatever policy is adopted with regard to stop lists, it is essential that the rules should be applied to search statements, but they need not be applied at the indexing stage. If they are applied to the index, this is only to reduce

its size and slightly to increase search speed. There are some arguments in favour of the inclusion of stop words in an index, together with the information that they are stop words.

An extreme example of a sparse stop list is that used in MELVYL: the title index stop list consists of:

A AND IN OF THE

and the subject stop list is

AND

THE OKAPI STOP LISTS

Okapi removes leading articles from title phrases where they were indicated in the original MARC records. So far as is possible, the same procedure is applied to search statements, by automatically removing

A An The Der Das Le La El

when one of these is the first word in a title phrase. It should be noted that, for example, **Il** cannot be removed unless the language is known to be Italian (not French) nor **Het** unless Dutch (and not English — though it seems rather unlikely there is a title **Het up in Samarkand**). There would be problems in dealing with words like **ein** and **un** which may be either an indefinite article or the number one. The fact that **thé** may be a French noun is unfortunate but unlikely to be serious in mainly English language files. A more intelligent system would carry a list of words which could be articles, and automatically search with and without, or issue a warning. It should be possible to cope with **L'**, but this has not been done in the prototype.

For word indexing, a single stop list is used, regardless of the source of the term. It consists of the eight most common function words from a 30,000 record subfile, which were, in order of descending frequency:

of and the in to for on an

(The word **a** does not appear in the list because words of one character had already been removed, and the frequencies of **the** and **an** are reduced because they had already been stripped where either of them was the first word in a title phrase.) (See Table 5.1.) The same stop list is applied to

search statements when a keyword search is to be done.

5.5.2 Authority files

If a machine readable authority file is available it can be used either as a form of index itself, or as an aid in the construction of an index. The latter method is used by the University of California's MELVYL OPAC: the Library of Congress Name Authorities file is used at indexing time to provide index keys for all forms of names which occur in the bibliographic records [5]. Whichever method is followed, the result is that a search for **Samuel Clemens** retrieves books by **Mark Twain**. (For libraries which do not have machine readable name authority files, indexing the MARC 900 cross reference fields may be a good substitute.)

In addition to name authority files, some libraries have machine readable subject heading lists which can be used in a similar way.

Okapi does not use nor make provision for the use of authority files.

5.6 Index storage and access

An index system consists of the *keys*, lists of *postings* — one or more lists associated with each key — and a means of rapidly accessing a given key, or of finding the "nearest" index key to a given search term.

5.6.1 Methods of storage

Keys may be stored *sequentially* in order (for example in alphabetical order), in *trees* or by *address calculation storage* (*hashing*). For a good discussion of data structures appropriate to the storage of indexes, see Knuth [6,7].

For fairly obvious reasons it is always easier to store keys in fixed units of memory, but given the intrinsically highly variable length of IR system index terms, particularly when some of the keys are phrases, the use of fixed memory allocation is impossibly wasteful of storage space (perhaps forty characters of memory would have to be assigned to each key, and the mean length of a key might be about twelve characters).

For interactive IR systems, the three most important considerations are (1) speed of access, (2) amount of storage and (3) ease of updating. These three aspects all conflict with each other. Sequential storage is quite fast,

needs the least storage space but is the worst for updating — the complete index may have to be rewritten to add one new term. Tree storage, which is very often chosen for IR systems of moderate size, is a very good compromise: speed is fairly good (possibly an average of about three disc accesses per search), there is a rather small structural storage overhead and updating can be done without rewriting the whole tree (although it may need to be pruned from time to time). Storing variable length data in a tree results in slower access and slightly higher overheads than for fixed length data. Hashing is extremely elegant and probably gives the best access times (although there is a trade-off between memory usage and speed); updating is almost trivial and memory usage fair. It has the serious disadvantage for OPAC use that it is not possible, with any conventional hashing technique, to locate keys in order. This means that a search of a hashed index can only return one of the two results **found** or **not found**. Since index terms cannot be retrieved in order, neither index display nor truncation by amalgamation of adjacent terms is possible. Hashing is ideally suited to the storage of keys which do not call for these facilities, such as copy numbers or control numbers. Some OPACs, including that of Cambridge University [8], use hash storage for their word indexes.

Choice of index structure for OPACs

The normal choice for free text IR systems accessing files in the range ten to a thousand megabytes would be either sequential or tree storage. Which method is chosen may depend on the speed and frequency with which the index may need to be updated. Some integrated library systems which use the same file of bibliographic records for cataloguing and for catalogue access offer immediate updating — a record is retrievable as soon as it has been entered. With either method of storage there is a penalty to be paid for this facility, taking the form of more elaborate structure and access procedures. Batch updating is in general quicker with a tree structure, but access is slower.

For files up to a few hundred megabytes which only need to be updated weekly or monthly a sequential index structure may be the best.

5.6.2 Postings lists

Whichever method of storage is chosen for the keys, each key has to be associated with its posting list — the list of source file records which are retrievable by means of the key.

Typically, each posting will consist of the number or address of a source file record and some information about the position and/or function of the key in the record to which the posting refers. It is easy to see that methods of storing and updating postings lists and of linking them to their index key also need some thought. One method is to store the posting list for a key physically adjacent to the key; this is simple and satisfactory for small files, but once a file grows to a size where there are many keys with more than a few dozen postings a sequential index would become unsuitable for browsing or truncation (in the sense of *stem-*searching), and with a tree index there are problems in construction and updating. Hence with medium to large files postings are stored separately from keys and each key is associated with a pointer to its first posting: before records from a successful search can be displayed, there must be one additional disc access to read the posting list.

5.6.3 Searching an index

Except for very small files (somewhat larger with very fast advanced disc systems), it is out of the question to search a sequential index sequentially. Normal practice is to use one or two levels of *directory* analogous to a set of volumes each of which has a thumb index: K - M is in volume six and **lead pollution** to **League of Nations** is on page 215. A search for **leadership** will cause page 215 of volume six to be read into internal memory from the disc. The page is then searched using either a sequential search or a binary search method. Obviously it is very simple to find adjacent keys, so browsing and stem-searching are easily supported. If the first level (volume index) can be retained in core it is possible to search an index of millions of terms with two or three disc accesses for each search (and at least one additional access to locate the first posting if the search is successful). Search time only increases very slowly with increasing index size (it doesn't take much longer to find a word in the OED than it does to find it in a pocket dictionary).

Tree indexes automatically provide the equivalent of the several levels described above, and the first level may be small enough to be retained in core. Variants of the *B-tree* structure described in Knuth [7] can store millions of keys using three or four levels, so a simple search can be almost as quick as with sequential storage; finding adjacent terms may need further disc accesses.

Searching a hashed index should need only one or two disc accesses, and only a small amount of core for the address calculation routines.

5.6.4 The representation of data in indexes

So far the discussion has applied equally to indexes containing any form of data, but the actual method of storage may depend on the nature and distribution of the keys — whether they are words and phrases, numbers or codes. It may be possible to reduce the amount of storage required by choice of representations of keys on disc. *Numerical* keys can often be encoded: ISBNs, for example, can be stored comfortably in five bytes, instead of the ten which are needed to hold them in character format. With *textual* keys, there are many forms of data compression which can be used. Many methods take advantage of the fact that the least quantity of information which can be stored is often eight bits, making 256 distinct character codes available, while the set of characters to be stored may contain as few as 37 distinct characters (A - Z, 0 - 9 and blank), leaving over 200 bit-patterns unused. The spare bit-patterns can be used to represent the most frequent digrams (letter pairs) or trigrams (triples). It is not difficult to obtain a storage saving of about a half.

Other methods take advantage of the ordered nature of the data. In a large index, most keys will have leading characters in common with their predecessors and successors. If at least the first two characters of a key are the same as those of the preceding key, there is no need to store the common characters: instead, the first byte of the key can say "the first N characters of this key are the same as the preceding key". Methods like this have been a good deal used in dictionary-type storage. They involve less processing than encoding methods which use look-up tables or extensive calculation. They do lead to complications in an index in which it is necessary to scan forwards and backwards. They are more valuable in phrase storage than in word storage.

5.6.5 One index or several indexes?

Most OPACs have several indexes, although a keyword system using reference retrieval software may not. Even if the index is physically a single sequence, practically all OPACs appear to the user as though there are several indexes: users have to choose an author search or a title search or a subject search; sometimes they can also choose an ISBN search or a series title search. Some types of search may access more than one index. The University of California's MELVYL has at least ten indexes. In MELVYL's simple "LOOKUP" mode an author/title search accesses

personal author, corporate author and title (i.e. title word) indexes, and a subject search uses subject heading word and title word indexes [5].

The number of indexes affects both the storage requirement and the speed of searching — in particular the number of disc accesses which have to be made. Unfortunately there are no clear cut rules for making this design decision. However the index or indexes are stored, it is essential that the system should include at least a certain amount of information about the source, in the bibliographic record, of a given occurrence of a given index term. If there are separate indexes, the word **Wood** as an author surname is automatically distinguishable from **wood** in a title or a subject heading, but the word has to be stored twice (at least). If, on the other hand, there is only one index, the term "wood" need only appear once, but there must be something associated with each posting which indicates what sort of beast it is. Another factor which influences decisions about the number of indexes is that, for structural and access reasons, it is not easy to include keys of very different types in the same sequence.

Examples of keys of very different types are *words* (**wood**), long *phrases* of fixed length in a form suitable for display as a brief record (**Hildreth, Charles R Online Public Access Catalogs: the User Interfa**) and control and accession numbers (**22.0041401**).

5.7 The Okapi index: storage, access and construction

5.7.1 Summary

The index consists of three files:

a *secondary index*, which is a single sequential list of index terms each associated with information about its source(s) in the bibliographic file and, in most cases, a list of postings

a *primary index*, used to access the appropriate area of the secondary index

a *postings file*, which contains the postings for frequently occurring terms

The secondary index is accessed (read from the disc) in fixed size "chunks" of two kilobytes. Each chunk contains an integral number of index terms and their associated data. No index term is split across a chunk boundary. Once a term has been located in the secondary index, the index can be scanned forwards or backwards from the term, without further access to the primary index. Terms are stored in the order determined by the collation sequence given in Section 5.3.4.

The primary index is held in core when the search program is running. There is one record in the primary index for every second chunk of the secondary index. Primary index records are condensed representations of the last term in the corresponding secondary index chunk.

Each posting consists of the 24-bit disc address of a record in the source file and six bits of information which associate the posting with up to six items of source information. If a term has ten or fewer postings, the postings are stored following the term within the secondary index; if it has more than ten they are stored in the postings file, and a pointer to the correct position in the postings file is held following the index term.

5.7.2 Choice of index structure for Okapi

It will be recalled (Section 2.7.2) that we started with two constraints: to minimise storage requirements and to minimise the number of disc accesses. The keys to be stored (words, phrases and class numbers) were all of types where both index display and truncation (stemming) facilities might be needed; this eliminated hash storage from consideration. The choice had then to be made between sequential and tree storage. Since Okapi was not envisaged as being part of an integrated library system, real time or very frequent file and index updates would not be required, so the choice fell on sequential storage on the grounds of simpler coding and smaller core requirement during searching. For the same reasons, it was also decided that only a single index sequence should be used.[1]

The disc storage constraint became less important after the PLAN 4000 LAN, with its larger disc, was acquired. The increased storage was used

(1) If Okapi were to be incorporated in an integrated system, the index should probably be redesigned as a tree structure. There are also some quite strong arguments in favour of the use of two or three separate indexes — a specific item index, a subject index and a librarians' index containing control numbers and copy numbers.

to provide more access points than had been intended. This is, in any case, in line with the trend towards cheaper mass storage.(1)

In conventional inverted file IR systems, indexing overheads (the amount of storage required for the indexing) can be as high as 400% of the storage needed for the source files, and are usually well over 100%. It was estimated that the minimum overhead compatible with the range of access points required for an OPAC would be around 60% to 70% of the source file size. This influenced the decision to use a single index sequence containing terms from all the indexed portions of the source file records. The prototype Okapi gave an indexing overhead of 78% on a file of 90038 records (see Table 5.3).

Since Okapi was not designed to handle files of more than a few hundred thousand records, it was estimated that it should be possible to store a sequential index as a two level structure, with the first level, the primary index, held in core memory at each user station. There must be a source file size at which it would be necessary to go to three levels of index. It is somewhat difficult to estimate this (it does not happen when the secondary index contains a certain number of terms — rather it is a matter of a trade-off between the number of chunks of secondary index which may have to be read and scanned to locate the sought term and the size of the primary index); certainly a file of a million records would need a three level structure. With a three level index searches would be appreciably slower, but there would be less core storage occupied by the primary index.

5.7.3 The secondary index

A chunk (two kilobytes) contains a sequence of variable length *index term records*. Each index term record consists of a pointer back to the beginning of the previous record (used when scanning the index backwards), the key itself, which is stored as a character string, and the number of postings which the key has for each beast (source type). These are followed either by a sequence of postings or (in the case of keys posted to more than ten records) by the address in the posting file of the first posting for the key.

(1) It seems likely that within the next few years many reference databases (ones which do not need to be updated very frequently) will be stored on digital optical discs, where very large amounts of data can be cheaply stored on one disc. It is not long since ten megabytes of magnetic disc storage was a large amount for a micro system. However, storage cost is still an important factor.

When a search is taking place a secondary index chunk is read from the disc and scanned sequentially from the beginning, because it is only the first index term record in a chunk whose position is known (it always starts in the second byte of the chunk). The sought term is not necessarily in the first chunk accessed: a few searches require the scanning of three chunks, and half the remainder require two to be read. The number of index term records in a chunk varies considerably with the region of the index being searched; the "low" region, which contains mainly Dewey numbers, but also dates of publication and titles which start with a digit, contains more terms per chunk than does the main alphabetical sequence. The overall average number is about 80 terms, so a search involves one, two or three disc accesses and a sequential scan of an average of perhaps 140 terms. The sequential scan is very quick compared with the time needed to do the disc accesses, and the difference increases as the number of users increases.

5.7.4 The primary index

Ideally, the primary level of the index should contain as much information as the labels on catalogue drawers. In order to know which drawer contains a given record (or would contain it if it were present), the drawer label must consist of the complete key of the last (or first) item in the drawer. To save one disc access per search, a complete copy of the primary index was to be stored in core memory at each user station. Since, with phrase as well as word indexing, the mean length of a key may be as much as twenty characters, and there may be many thousands of "drawers", twenty-character primary index keys would need a prohibitively large amount of storage. The compromise solution which the team evolved is described below.

It was estimated that, with careful programming, it should be possible to have up to about 12 kilobytes of primary index resident in core at the same time as the search program and enough storage for input/output buffers.

Each primary index record is generated from the last term in a secondary index chunk. (There is no particular virtue in using the last term, the first would do equally well.) To maximise the amount of information in the primary index, that is, its discriminatory power, a minimum requirement is that all or at least most primary index entries should be unique (this is less stringent than the requirement that index entries should

contain full "catalogue-drawer" information). It had been hoped that enough discriminatory power could be achieved simply by taking the first four characters — the initial tetragram — of the last term in a chunk to be the primary index record corresponding to that chunk. Four-character entries were chosen because three are clearly not enough for any but the smallest files, and four permits a simple form of order-preserving data compression.

The discriminatory power of initial n-grams depends on the number of terms in the secondary index and on the secondary index chunk size (or rather, the average number of terms in a chunk): the more terms there are in a chunk, the more likely it is that the last terms in successive chunks will differ in the first n characters. It also depends on the relative frequencies of different initial n-grams in the language of the index file: there are more words beginning with **ELEC** than with **AARD**. The situation is further complicated when the same index contains items of different source types. Non-fiction title phrases tend to begin with broad subject descriptors or conventional adjectives — "**economics of ..**", "**brief history of ..**". Subject heading phrases also start with broad terms. Personal names have their own idiosyncracies ("**Mac..**"), but do not cluster so seriously as do phrases.

Experimental indexes were constructed from subsets of the source file of up to 33,000 records. At the 33,000 record level, there were a substantial number of secondary index chunks which did not differ in the first four characters of the last term. The worst case was "**ECON**", which filled eight chunks. There were few cases of more than two repeats, and it was felt that this structure would just be acceptable for a file of this size. However, the tetragram key would clearly not be acceptable for the full 90,000 record source file. To check this, the complete file was indexed (producing about 370,000 distinct index terms — see Table 5.3). Table 5.4 shows the distribution of the initial tetragram keys for this index.

A hybrid structure was devised in which tetragram clashes are resolved by using a form of run-length coding and, where necessary, a pointer to a "tail" record in a separate overflow area of the primary index. This could be applied to indexes of any size, but the "tail" storage requirement might become rather large. For the 90,000 record file, with its 370,000 distinct index keys, the tail section of the index is only 612 bytes (about 5% of the total primary index size). The method of searching this primary index is a little more complicated than a simple binary chop. The

structure does not quite contain full "catalogue drawer label" information, but it guarantees that not more than two "drawers" need to be scanned. (A "drawer" here is the equivalent of two chunks, since a primary index record addresses a pair of chunks.)

5.7.5 Postings and beasts

Each posting is 32 bits long, and consists of a 24-bit source record address, and a single byte, six bits of which are used to refer back to any combination of up to six of the ten possible source types (which we refer to as *beasts* —see Section 5.2.3). (The two spare bits are used to hold some structural information which is used by the search and term combination routines.) Thus a given index key can occur as up to six beasts. We tried to think of examples of character strings which might come from more than six different sources. "**1984**" is a possible candidate: title phrase, title word, date of publication, Dewey **198.4** (unallocated, as it happens), subject phrase and word, corporate name word. One indexing run failed on "**Aristotle**", who turned out to be both personal and corporate and has the unfortunate attribute of constituting both a phrase and a word when he occurs as corporate name, title or subject. By the time the prototype stage had been reached corporate and personal name beasts had been merged, as had corporate name words and surnames. However, **Aristotle** does still occur in six roles: AU, AW, TPH, TW, GS and GP (these abbreviations are defined in Section 5.2.3).

The 24-bit record address in a posting allows access to 64 megabytes of storage, or about 300,000 records of the size used in the Okapi prototype (Chapter 4).

Postings for relatively uncommon keys, which would include almost all title phrases, are stored in the secondary index itself, adjacent to the term to which they belong. The figure of ten, above which postings are stored in a separate file, is fairly arbitrary.

5.7.6 Construction of the Okapi index

The index is constructed by running a suite of three programs. The first two can be run in parallel at a number of stations on the network, but for simplicity the following description assumes that only a single station is being used.

The first program (key generation) operates by reading the source file sequentially and extracting and generating the appropriate keys from each record. When internal memory is full the keys and their associated postings are sorted and merged in core, and output to a temporary file. Each batch of sorted and merged keys is referred to as a "run". A run contains typically about 2000 keys. The process is repeated until the entire source file has been read, the result being several hundreds or thousands of runs of keys stored in one or a few temporary files. This program makes use of the procedures for key generation and construction described in Sections 5.4.1 – 5.4.4.

The second program (merge) performs a series of simple merges, each of which combines up to ten runs into a single, longer run of keys and their postings. Each "pass" of the temporary file reduces the number of runs by a factor of ten until finally there is a temporary file containing one single run of index keys, each followed by all of its postings. A file of the size on which the prototype Okapi operates needs three or four passes.

The final index production program takes the raw output from the merging operation and generates from it the three index files described in Section 5.7.1. This program could be described as the one which puts the structure into the index, the raw material for which was produced by the key generation program.

References

1 **Harris J L.** Alphabetical arrangement: some unexamined issues. In: Proceedings of the 42nd American Society for Information Science Annual Meeting, Minneapolis, Oct 14-18 1979, *Information choices and policies.* Knowledge Industry Publications, Volume 16, 1979, p214-218.

2 **Larson R R** and **Graham V.** Monitoring and evaluating MELVYL. *Information Technology and Libraries 2* (1), 1983, p93-104.

3 **Mischo W H.** *A subject retrieval function for the online union catalog.* Technical Report. OCLC, 1981.

4 **Jones K P** and **Bell C L M.** The automatic extraction of words from texts especially for input into information retrieval systems based on inverted files. In: *Research and development in information retrieval.* Proceedings of the third joint BCS and ACM symposium. King's College, Cambridge, 2-6 July 1984.

5 **Radke B S, Klemperer K E** and **Berger M G.** The user-friendly catalog: patron access to MELVYL. *Information Technology and Libraries 1* (4), 1982, p358-371.
6 **Knuth D E.** *The art of computer programming. Vol 1: Fundamental algorithms.* Addison-Wesley, 1968.
7 **Knuth D E.** *The art of computer programming. Vol 3: Sorting and searching.* Addison-Wesley, 1973.
8 Cambridge University Library's catalogue takes on a new life. *VINE 47*, 1983, p4-7.

Table 5.1. The most frequent terms in a file of 30,081 MARC records

The list was obtained by extracting words from the title, name and subject fields of the complete holdings of a Polytechnic site library. Initial definite and indefinite articles had already been removed, and one-letter words were ignored. A word is only counted once per record even if it occurs in more than one field.

Term	Freq	Term	Freq
of	13576	political	1346
and	11533	international	1267
the	8815	society	1214
in	7914	development	1181
social	3251	economics	1173
to	3194	library	1152
britain	2652	management	1150
great	2404	politics	1147
for	2387	modern	1108
series	2345	aspects	1083
economic	2146	research	1060
on	1951	university	1038
studies	1921	sociology	953
history	1584	theory	951
british	1497	education	940
an	1381	policy	901
study	1358	american	896

Evidently it is a humanities and social sciences library.
The high frequencies of **study/ies, series, library, research, university** and some other terms are a consequence of indexing words from series titles.

Table 5.2. Analysis of index terms by type of key — file size 30,081 records (the complete holdings of one Polytechnic site library)

Type of key	Number of distinct terms	Mean no of recs per term	Mean terms per rec	Mean term length
title phrase	32841	1.3	1.4	32.0(1)
title word	18618	12.2	7.5	8.0(2)
pers name(3,5)	20441	1.7	1.2	9.6
surname(4,5)	11390	3.1	1.2	6.9
corp phrase	1479	2.4	0.1	33.3(1)
corp word	1805	9.8	0.6	7.9(2)
subj phrase(6)	9261	3.6	1.1	21.4
subj word(6)	6060	10.7	2.2	8.0(2)
class number	9425	3.3	1.0	8.8
total	99770(7)		16.3	

Notes

(1) This is the mean length after truncation, where necessary, at 40 characters.
(2) The mean word length is high because the mean is taken over types, not tokens. Mean token length was not measured, but could be expected to be between five and six characters.
(3) Surname + initial(s), separated by blanks.
(4) First element of surname only. 8620 (76%) of the surnames were associated with a unique set of initials.
(5) Excluding names from subject headings.
(6) Quite a large proportion of records have no subject headings.
(7) Approximate total number of distinct terms.

Comments

The marginal number of new index terms per record is 3.6 — i.e. each new record contributes, on the average, 3.6 new terms. Most of these are title phrases and personal names. Because of the mixture of data types it is very difficult to predict how this figure will decrease with increasing file size. The rate of increase of the number of *words* with file size is already low (about 0.6 words per record) even in so small a file. On the other hand, many title phrases and personal names are unique.

Table 5.3. Index size and term occurrence statistics (file size 90,038 records). (From the inversion of the complete Polytechnic catalogue)

Number of terms	Number of records
274995	1
42949	2
16647	3
8644	4
5497	5
3793	6
2646	7
2077	8
1539	9
1300	10
12766	more than 10
372853	

Total number of postings: 1515000 or about 16.8 per record

Marginal number of new keys per record: 4.1

Mean number of keys stored per chunk of secondary index: 69.8

Mean number of characters per key: estimated between 14 and 17

Secondary index size: 10680 kilobytes
Primary index size: 11 kilobytes
Postings file size: 3739 kilobytes

Total index size: 14420 kilobytes = 77% of source file size

Table 5.4. The most frequent initial tetragrams (file size 90,038 records). (From the inversion of the complete Polytechnic catalogue)

4-gram	rank	4-gram	rank	4-gram	rank
inte	1	comm	9	arch	13
				euro	
soci	2	cont	10	guid	
		new		nati	
intr	3	stud		stat	
				the	
poli	4	elem	11		
		indu		appl	14
comp	5	theo		coll	
				dict	
econ	6	amer	12	hand	
		cons		lang	
brit	7	engl		plan	
		grea		stru	
hist	8	mana		tran	
mode		prin		univ	
				work	