6 Design and implementation

6.1 Introduction

This chapter describes the retrieval devices to be evaluated in this project. These are:

stemming and spelling standardisation

the use of a lookup table of cross references and "go" phrases

a method of suggesting corrections for words which may be misspelt

The devices were used in two catalogues referred to as *EXP* and *CTL*. *EXP* incorporates all the devices, including two-level stemming. *CTL* has "weak" stemming, but none of the other devices. The catalogues appear identical in casual use.

The catalogue systems are described here from the point of view of structure and implementation rather appearance. The appearance and presentation of the catalogues is described in Chapter 7. Some may prefer to read this chapter and Chapter 7 in parallel.

This chapter also describes the method of term combination which we used. This is combinatorial rather than boolean. Most post-coordinate retrieval systems either use explicit boolean AND and OR or, more frequently in online catalogues, an implicit boolean AND. Okapi systems, like CITE (2.5) and the "keyword" subject enquiry in the SWALCAP LIBERTAS system, retrieve all records which contain enough of the words in the user's search, outputting the "most similar" records first.

6.2 Stemming and spelling standardisation

6.2.1 Background

Automatic stemming has been investigated many times for reference retrieval systems but is only used in a few online catalogues (see Chapter 3 for a survey). In reference retrieval systems most researchers have concluded that it is beneficial. It appears to increase recall without unduly decreasing precision, and its performance is said to be comparable with that of manual truncation by intermediaries. When stemming has been used in online catalogues these have usually been catalogues which access

specialised collections. The prime example is the National Library of Medicine's CITE [1]. We do not know of any online catalogue which uses stemming in accessing a general collection for general users.

It seems likely that one of the reasons for the apparent success of automatic stemming in reference retrieval systems is that the searcher is generally trying to retrieve one or more sets of records for printing offline. These output sets are usually produced by boolean combination of a number of intermediate sets. It is perhaps not too much of a generalisation to say that the searcher (intermediary) seeks to construct a canonical formulation of a search. It may not matter if some of the intermediate sets or single term searches contain a substantial proportion of false drops attributable to stemming.

The use of general online catalogues is very different. There is no need, and rarely any attempt, to find a "definitive" search statement. Most search statements consist of only one or two words (see, for instance, Table 8.3, which shows that the most frequent number of terms is two, with a mean of about 2.2; Okapi users are quite typical in this respect). A session at the catalogue often consists of a sequence of related searches, any of which may contribute to the user's satisfaction or lack of it.

When automatic stemming is used on searches consisting of only one or two words even the most conservative procedure often gives false drops which would have been ANDed out in a longer search. Even conflating singular and plural forms can lead to a good deal of noise: examples drawn almost at random from Okapi '84 transaction logs are *right, rights; age, ages; lord, [house of] lords; mass, masses; art, arts; account, accounts; communication, communications.* The removal of "ing" and "ed" endings can also be detrimental: *age, aging, aged; account, accounting; market, marketing; train, training.* An examination of any word list drawn from a collection of noun phrases shows that other suffixes such as "er", "tion", "ence", "ism", "ist", "..ity" often affect the meaning very markedly, and that their removal can be expected to lead to a significant proportion of false drops. Most stemming evaluation has been done on searches of databases in the "hard" sciences.

We looked at 6700 consecutive subject searches (after removal of immediate repetitions of a search) from the logs of an Okapi '84 terminal to try to get an idea how often stemming would be detrimental and how often beneficial. The types of word looked at were regular English plurals, "ed" endings, "ing" and "ings" endings, and "tion" and "sion" endings and their plurals. The results are summarised in Table 6.1.

After removal of very common function words and stop words,

these search statements contain 3585 distinct words (*types*) and 14,584 occurrences of the words (*tokens*), a mean of 2.2 words per search. (For *all* tokens, the mean number per search statement is about 2.6.)

Table 6.1 Types of word used in subject searches (Okapi '84)

	Types	Tokens	Searches
regular plurals	715 (19.9%)	2201 (15.1%)	
'ed' endings	56 (1.5%)	125 (0.9%)	
'ing(s)' endings	177 (4.9%)	638 (4.4%)	
'tion(s)' and 'sion(s)' endings	255 (7.1%)	1107 (7.6%)	
nonce-words (excluding numbers)	1833 (51.1%)	1833 (12.6%)	
misspellings and rubbish	565 (15.8%)	650 (4.5%)	
Total	3585	14584	6692

The table shows that plurals are far less frequent than singulars, that "ed" endings are probably rare enough to be disregarded, but that some other suffixes occur quite often.

Any stemming procedure has to be applied both to the index language and the search language. The above breakdown could be usefully compared with a corresponding table compiled from the language of bibliographic records, with title words and controlled subject headings treated separately. Unfortunately we have not had time to do this. We would expect to find that the distribution of suffixes in search statements is quite similar to that in titles, but markedly different from their distribution in subject headings.

Because subject headings are for description and recognition rather than for searching, they have their own "subgrammar" One feature of this is an extensive use of plurals where searchers and authors would use singulars. (For an enlightening and depressing discussion of Library of Congress subject heading practice see Mischo [2].

Schabas [3] found that the performance of PRECIS was similar to LCSH when used for keyword searching.)

6.2.2 Functional design considerations

Mitev and Walker [4] asserted that the first records shown should be those (if any) containing the exact phrase entered by the user. The search system should then, if necessary, look for records which *partially match* or *bear some resemblance* to the search statement. They hold that a search should be automatically broadened when appropriate (user is not satisfied, no records found, only a few records found) by any or all of the following means:

1 stemming

2 relaxing the word order constraint

3 using word fragments

4 looking for records which contain only some of the words of the search.

In this research we were not concerned with word order, except in so far as one of the catalogue systems (*EXP*) uses a small dictionary of phrases. The Okapi indexes do not contain adjacency information, and Okapi suffers from the usual "false coordination" effects. (Try LIBRARY SCIENCE or WAR AND PEACE on any catalogue which does an implicit AND on title and subject words.) Nor do we use word fragments. See Chapter 4 for a discussion of the use of n-grams for finding classes of morphologically similar words.

The fourth heading - being prepared to display records which may not contain *all* the words of the search statement - is a feature of CITE, Okapi and of the SWALCAP LIBERTAS system. Again, it is one which has been investigated in reference retrieval systems with generally favourable results. It is not the subject of this project, although there is a strong connection between the way we used stemming and the way in which terms are combined or "merged".

This merge procedure is described in 6.5.

6.2.3 Strong and weak stemming

Ideally the system should first look for records which match the actual words of a search, without any stemming. This would be followed by "weak" stemming (conflation of plural, singular and "ing"), and this in turn by "strong" stemming. The degree of stemming, if any, to be applied initially might be made to depend on the number of words in the search and possibly on their frequency in the index. For example the search SUCCESS AT INTERVIEWS finds no

matches in the PCL file when records containing "success" and "interviews" are sought, and still fails when plurals and singulars are conflated. But when stronger stemming removes the suffix from "successful" it finds a book with the title

> "How to succeed at an interview : .. a guide to successful preparation for job applications .. ".

(JOB APPLICATIONS is a better way of expressing this search. It finds 11 books, all but one of which are likely to be relevant. Having found the above entry to the file a user might decide to try this reformulation of the search.)

Another example is TERMINAL ILLNESS which finds no records without stemming, but the file contains four relevant records indexed under "terminally" and "ill".

Any catalogue which does not find these records in response to these search statements is, to put it mildly, inadequate. Few users would try rephrasing the last example as THE TERMINALLY ILL.

6.2.4 Spelling standardisation

Many common words have alternative acceptable spellings. Most of these represent differences between British and American English. Even within British English there are *judgment/ judgement*, *connexion/connection* and many others.

A large proportion of the differences can be given in the form of rules rather than tables, although most of the rules ought to have exceptions. Replacing every *iz* by *is* makes a few words such as "size" look rather odd. This is one of the reasons why we decided not to show the user how the system represented search words. We had to design the interaction so that links are maintained between what the user types and what the system is processing.

Rare variations ("ium" = "um" at end if word is "aluminium") are handled by lookup tables (6.3).

Since the spelling standardisation was to be rule-based we decided to incorporate it in the stemming procedure. The rules were concocted after a study of past searches on Okapi, and of lists such as that given by Paice [5, p86].

6.2.5 Two levels of stemming

There are obvious examples (*right*, *rights* etc) which show that it is not always safe to apply any stemming at all, but for reasons of simplicity, economy and ease of evaluation we chose to apply a "weak" transformation, which included the spelling standardisation, to all searches. We hoped that few searches would be seriously affected by the

unconditional conflation of plural/singular and 'ing' forms.

We then had to decide how to sequence the application of the two levels - should the second level only be applied if requested, or if the first did not lead to any retrievals containing all the stems in the search?

The method used in Okapi '86 is extremely simple, but could not be applied in a system which (like most current online catalogues) does an implied boolean AND on the words of the search.

Both weak and strong stems are generated and looked up in the appropriate indexes. Each strong stem is marked as being 'synonymous' with its corresponding weak stem. Each stem is assigned a weight based on the inverse of its frequency, so common words are worth less ('lighter') than rare ones. In particular, a strong stem generally has a lower weight than its corresponding weak stem. The posting lists for all the stems are then merged, each posting in the output list being given a weight which is the sum of the weights of the contributing stems, except that the weight of a strong stem is not added if the corresponding weak stem has contributed. This avoids giving falsely high weights to records which contain, for example, both 'success' and 'successful'. (See 6.5 for a fuller description of the merge procedure. The procedure described above is only used in the experimental (EXP) system - the control (CTL) system only uses weak stems.)

Additionally, postings are not accepted for the output list unless their weight reaches a certain threshold based on the weight of a hypothetical record containing all the weak stems in the search. This ensures that records bearing little resemblance to the search are not retrieved.

6.2.6 Interaction design

How should the stemming be presented to the user? Should the system explain what it is doing? If so, how?

Many users do not know that a keyword catalogue looks for 'words' rather than 'ideas' or 'concepts'. (Some of those who do realise roughly what is going on regard it (rightly) as 'unintelligent'.) The catalogue should give some introductory explanation of how it works, but this must be very brief and pithy. It is undesirable to present (except on request) more than an absolute minimum of non-essential text since it is impossible to make people read the screen. Two lines of explanation is about the maximum.

Examples:

> This catalogue looks for items containing the words you type in and other words similar to them
>
> The computer will look for books whose TITLES and SUBJECT HEADINGS contain as many as possible of the words you type

Both examples are intended to suggest that the computer looks for words rather than phrases. The first is about as far as one can go in casually introducing the idea of stemming. The second can be used for "best match" systems such as Okapi which do not do an implicit AND. We have not been able to think of any message which is short enough and introduces both features.

One function of explanation is to try to avoid users being put off when a search fails or retrieves unexpected items ("That isn't what I asked for!"). When a search fails (to find anything) the system should be able to give some sort of explanation at that point, but it cannot - by definition - know when it has displayed false drops. It is not so much to excuse the system but to reassure the unsuccessful user. It is important that a catalogue should not often make a fool of *itself*, but it is more important that the catalogue should not make a fool of the *user*.

Another function of explanation is to help users improve their search techniques. The fact that the system is searching for *words* rather than *concepts* is more likely to be appreciated if it displays each word as it is searching for it. Further, if the search should fail, some users may have noticed which words are likely to be useful and which are not: this may help them if they have to rephrase their search.

Presentation during searching must depend partly on the output from the stemming and spelling standardisation. If we had used a procedure which performed "normalisation" on the stems it produced, so that they are always "real" words, it would be possible to display the stemmed words to the user. For example, if "computing" weak stems to "compute" and "computer" and "computation" strong stem to "compute" the system could display

> Looking up "compute.."

or even

Looking up "compute"

followed by

Found 2317 books under "compute" and similar words

but one cannot display "comput", "censu" or "filosofi", which are forms produced by our stemming and spelling standardisation. (We also tried using plus signs to indicate truncation. The only concise and transparent ways of suggesting that a word or phrase has been truncated, or that some of it is "missing", seem to be two or more dots or the word "etc".)

6.2.7 Choice of stemming procedure

We used a version of Martin Porter's suffix removal algorithm [6] because it is short (easy to code and does not need much memory), tends to "under-stem" and has been shown to perform well in reference retrieval searching [7, 8].

This algorithm removes apparent "s" plurals followed by "ed" and "ing" from words of any length, and then successively removes other suffixes if the remaining stem would be long enough (the measure of length is a quasi-syllabic measure based on the number of vowel-consonant transitions). Most final double consonants are reduced to single, so that "travel" and "travelling" etc are properly conflated.

We adapted Porter's procedure, making it a two stage process and adding spelling standardisation in Stage 1.

The Stage 1 process is intended to be non-contentious - that is, its application should rarely affect the meaning of a search. It is essentially Step 1 of the Porter algorithm with spelling standardisations. We refer to this as "weak stemming". The second stage involves carrying out Steps 2-5 of the original Porter algorithm; this we call "strong stemming".

Thus any word has both a weak and a strong stem: the weak stem is considered "closer" to the original word.

6.2.8 Stage one - weak stemming and spelling standardisation

This stage removes regular English plurals and "ed" and "ing" endings, then reduces most double consonant endings to single. No endings are removed from words under four letters long or from "words" which contain digits or other non-alphabetic characters (hyphens have already been squashed and/or replaced by blanks).

There is an exception table containing one entry: the word "united" (United States, United Kingdom etc.). Conflating "united" with "unit" leads to false drops.

For other words, Step 1 of the original Porter algorithm is done, followed by these spelling standardisations - an example follows each. In many cases they cope with differences between American and British spellings of words; in some cases they trap common spelling errors.

1 iz --> is
(organize --> organise)

2 ae --> e except at the end of a word
(orthopaedic --> orthopedic)

3 ph --> f
(sulphur --> sulfur)

4 oe --> e
(foetus --> fetus)

5 our --> or if word length greater than five
(behaviour --> behavior)

6 exion --> ection at the end of a word
(connexion --> connection)

7 nse --> nce at the end of a word
(defense --> defence)

8 amme --> am at the end of a word
(programme --> program)

9 gue --> g at end
(catalogue --> catalog)

10 ism --> ist at end
(feminism --> feminist)

11 ant --> ent at end
(dependant --> dependent)

12 tre --> ter at end
(centre --> center)

13 anc* --> enc* at end if length > 6 (* denotes one letter or no letters)
(dependance --> dependence)

For many of the changes it doesn't really matter which way round they are done: for example we could equally well do "ent" --> "ant" at the end of a word. It is more a case of making words equivalent.

6.2.9 Stage two - strong stemming

This is the same as the original algorithm Steps 2 - 5, with one or two minor changes made necessary by the altered Stage 1, mainly that 'iz' is always 'is' and 'ism' at the end of a word is always 'ist'.

This removes, if the word is long enough, endings such as 'sion', 'tion', 'ence', 'ous', 'ness', 'ate', 'er', 'ic', 'able' and 'ible', 'ive', 'ize', 'ism' and 'ist'. (For a complete description see [6].)

6.2.10 Discussion and examples

We introduced the idea of two-stage stemming because drastic stemming applied to searches containing only one or two words often leads to false drops. A good example is 'communism' and 'communications' which both strong stem to 'commun'. Another example is 'organisation' and 'organic'. When searching for words we search for both weak and strong stems, but the weak stems are given a higher weight than the strong ones.

We have tried to take account of common spelling variations, mainly differences between British and American spellings. We also try to cope with words with more than one acceptable spelling, e.g. *archaeology, archeology; foetus, fetus; mediaeval, medieval*. The changing of endings 'ant' and 'anc*' to 'ent' and 'enc*' respectively is because users often make mistakes with such endings, and where either ending can be correct the words are similar enough in meaning to justify treating them the same. (Examples are *independance, independence; correspondent, correspondant*.)

Words often look rather strange when weak or strong stemmed. Okapi uses stemming in its index and then again on the user's search statement, but these stems are never actually displayed to the user, so their appearance does not matter. For example, we get 'fotografi' from 'photography', but the former only serves as Okapi's internal representation of the term: as far as the user is concerned the computer is searching for 'photography'.

Spelling standardisation often acts on words in unintended ways. Again, in most situations this doesn't matter. The word 'upheaval' stems to 'ufeaval' - but as long as no other word stems to 'ufeaval' it will cause no problems. Some other unintended alterations are:

lenses	lence
resource	resorc
aeroplane	eroplane
uphill	ufill

dizzy	diszi
advance	advence

In practice, none of these (nor many others) matter, but there are a few which should be treated as exceptions, like "united" (see *Some oddities* below).

We decided that record retrieval would be improved by treating "ism" and "ist" at the end of a word as equivalent. Examples include *socialism*, *socialist*; *Marxism*, *Marxist*; *structuralism*, *structuralist*; *racism*, *racist*; *fascism*, *fascist*.

6.2.11 Some oddities

Inevitably some odd things happen with stemming. Some of these happened with the original algorithm but others are a direct consequence of our adaptations. These are a few of the transformations which may decrease precision or even recall:

herring	her
organism	organist
poetry	petri
poet(s)	pet
shoes	she
schism	schist

And place names do not always lend themselves readily to stemming:

Woking	woke
Dungeness	dung

6.3 Phrases and the *go/see* list

The Okapi *go/see* list contains classes of terms which we treat as equivalent, and phrases which we treat as single terms. It was compiled after study of a large number of transaction logs of the use of Okapi '84 in Riding House Street library, i.e. the emphasis is on the words and phrases that are most likely to be searched for in Okapi.

An "equivalence class" of terms is a list of words (or phrases) which we treat as having the same meaning at search time. For example, "France, French" may comprise one class, and "VDU, Visual Display Unit, VDT, Visual Display Terminal" another. A special case is when the list has only one item, which will be a phrase we want to treat as a complete phrase rather than as individual words. Examples of this are "industrial revolution" or "soap opera".

In the index, any member of a given equivalence class will point to the same set of postings, which will be the set of all postings which contain at least one of the class terms. Thus a search for "France" may retrieve records which contain the word "French", and *vice versa.*

6.3.1 Categories of equivalence classes

The classes in the *go/see* list may be roughly categorised into the following eight groups, although some classes belong to more than one group.

1 Abbreviations

"BBC, British Broadcasting Corporation"
"CND, Campaign for Nuclear Disarmament"
"TV, Television"
"VAT, Value added tax"

2 Noun/adjective pairs - mainly for proper names

"Switzerland, Swiss"
"Wales, Welsh"
"Florence, Florentine"
"Freud, Freudian" etc.

3 Alternative terminology

"Russia, USSR, Soviet Union, Russian ... (etc.)"
"movies, moving pictures"
"Holland, Netherlands, Dutch"
"Conservative Party, Tory Party"

Included here are terms which may be one word or two words

```
"micro computers, microcomputers"
"ultra violet, ultraviolet"
"infra red, infrared"
"Serbo Croat, Serbocroat"

4  Alternative spellings

"gaol, jail"
"csar, czar, tsar, tzar"
"aluminium, aluminum"
"aeroplane, airplane"

5  Common irregular plurals

"man, men"
"woman, women"
"phenomenon, phenomena"
"wife, wives"

6  Related terms

"child, children, childhood, childish"
"tax, taxation, taxability, taxable"
"Third World, underdeveloped countries, developing
 countries, .. "

7  Numbers

  "6, six, sixth, vi"
(and similarly for other numbers up to 20)

8  Phrases

"Pirate Radio"
"Labour Party"
"Official Secrets Act"
"Notting Hill"
"General Strike"
```

6.3.2 *Phrases*

The use of phrases poses some problems, not least of which is the question of what to include. The main use of phrases is in keeping together words which would lead to false coordination if separated. However, care must be taken in selecting phrases for the list because the inclusion of a phrase in the *go/see* list can sometimes lead to a reduction in the number of relevant books found.

The classic example of a "good" phrase to have on the list is "soap opera", because books on "soap" alone or "opera" alone would not be relevant in such a search. In practice, a search for just "soap opera" would retrieve relevant books even without the phrase list, but as soon as any

other terms are included in the search there is a danger of words becoming separated. For example, a search for "The production of soap operas" could equally well retrieve books on "soap production" or "production of operas", neither of which, we assume, would help the user much.

Similarly, in most other cases, there is little to be gained from using the phrase list if the phrase is the only thing in the search. But when other terms are added the presence of phrases in the index helps to prevent false coordination.

WHICH PHRASES?

A phrase whose constituent words have meanings very different from that of the whole phrase is an ideal candidate for inclusion in our list. However, most cases are not as clear-cut as "soap opera". In many instances some or all of the component words are related in some way to the phrase meaning. For example, consider the phrase "pirate radio" - books on "radio" alone might be relevant, whereas books on "pirates" almost certainly would not.

Phrases which contain very general words should be included because they provide ample opportunities for false co-ordination. "Welfare State", "middle class", "General Strike" and "social science" would come into this category. A search for "Middle class education" could retrieve books on "education in Middle schools" or "education in mixed ability classes". Similarly a user requesting "Women and the Welfare State" might be offered a book on the welfare of women in New York State. If we can trap known phrases such as the above, we will reduce the probability of these false drops.

We included some political parties ("Labour Party", "Liberal Party", etc.) because "Party" can easily attach itself to the wrong word. A search for DECLINE OF THE LABOUR PARTY retrieved "The decline of the Liberal Party 1910-1931".

Some phrases should not be included in the *go/see* list: they are phrases which can be (and are likely to be) expressed in alternative ways in the records. For example, "French Revolution" might be considered suitable for inclusion in the list. Certainly this would lead to all books on the French Revolution being retrieved if this topic was searched for. However, what would not be found are all the equally relevant books with titles or subject headings such as "Revolution in France", "The French and Russian Revolutions", or "France during the revolution", etc.

With all phrases there is some danger of the above situation arising. For example, if we include the phrase

"World War II" then we would miss books on "World Wars I and II". However, in such a case the advantages of including the phrase probably outweigh the disadvantages. With any phrase, the decision whether or not to include it must be a fairly subjective one, and we must rely on analysis of users' subject searches to measure the usefulness or otherwise of our various choices.

6.3.3 Problems in searching for phrases

At search time, we always scan the index for the longest match possible with the user's search string, thus picking up phrases where they occur. A "term" in the search may therefore be either a single word or a phrase. When the postings are merged, inverse frequency weights are assigned to the terms. At the moment we have no special way of assigning weights to *go/see* list terms, and their weights may therefore be a little artificial.

There is a danger, for example, of terms being given artificially low weights because they belong to an equivalence class and are thus highly posted. Because the list of postings for "Spain" also includes all postings for "Spanish", "Hispanic" and "Espana", the weight assigned to "Spain" in a search will be lower than it would be if "Spain" were not on the *go/see* list. In most cases this will probably not have disastrous consequences but it is something which should be borne in mind, perhaps when compiling the *go/see* list.

For the above reason, there seems to be a case for simply adding 1 to the weight of any *go/see* list term. In the case of phrases, a suitable way to assign a weight might be to calculate the weights of the component words, sum them, and then add 1. It might also be a good idea to actually add the individual words to the merge, because the weighting system just described would ensure that postings containing the actual phrase would have a higher weight than those only containing the individual words, with the added advantage that phrases like "French Revolution" and "World War II" would work better (see discussion above).

A disadvantage of applying techniques such as the above is that it makes the (already complex) search string analysis even more complicated, and involves more index searching which takes a relatively long time.

6.3.4 Other go/see list categories

We don't include abbreviations which are in themselves words, such as WHO or AIDS, since this could lead to some very strange retrievals.

Other equivalences are particularly useful for rare terms because relevant books will often be found which would not

otherwise have been. For example, there are very few books on Tibet, and all contain only one of the terms "Tibet" and "Tibetan". Thus by making these two terms equivalent we can retrieve all the books on Tibet rather than just those containing the word "Tibet". A search for "Tibetan religions" will find "Politics and Religion in Tibet".

6.3.5 *A note on stemming and indexing*

The index contains both "weak" and "strong" stems of words (6.2.3). Originally, we intended that if a phrase was on the *go/see* list, only the strong stems of its component words in the index. However, there turned out to be serious problems with this approach and so now all words go in the index as both weak and strong stems regardless of whether they also form part of a phrase.

The reason for the problems with the former method is that when a phrase's component words are relevant in themselves, they get "lost" if searched for on their own. For example, the phrase "Communist Party" is on the *go/see* list. Originally, this meant that although any books containing "communist party" would be posted under this phrase, they would only be posted under the strong stem of "communist". Thus they would only receive a low weight in a search such as "Communist politics" where they would be very relevant.

6.4 Spelling correction

To undertake spelling correction interactively (word-by-word) on limited hardware it is essential to use a two-stage process (Chapter 5).

The first requisite is a fairly large dictionary - large enough to contain the majority of words which users may misspell. Some systems store several categories of words: there may be a special dictionary of very common misspellings linked to their correct forms (e.g. ADN --> AND, NECCESARY (etc) --> NECESSARY). This is the only practical way of correcting short words. Sometimes the dictionary is coupled to a set of rules for "expanding" entries by adding plausible affixes (EXAGGERATEDNESS might not be in the dictionary, but could nevertheless be accepted as an allowable word). The dictionary may also be partitioned into specialised sections to be used in different contexts.

The second requisite is a fast lookup procedure for selecting dictionary entries which are near enough to the user's word to be considered as possible corrections. The lookup should return a list which balances the requirements of recall and precision. It should include all or almost all candidate replacements, but must be of manageable size because each word in the list is going to have to be matched against the user's word to produce either a single word for automatic correction or a short list for the user to select from.

Finally, there is a matching algorithm which measures in some realistic way the likelihood of a given candidate word being a proper correction for the user's word.

6.4.1 Dictionary

The obvious source for the dictionary is the bibliographic file itself.

We first tried extracting all words from the subject-rich fields of the bibliographic source file. Since it is dangerous to try to correct short apparent misspellings, we selected only words of five letters or more, and excluded all numbers and "words" containing digits. There was a considerable proportion of non-English words. Experiments showed that such a system would sometimes suggest correction to a foreign word. Since our users only very rarely enter non-English words (other than proper names) we tried to eliminate as many as possible by not extracting title words from records whose MARC 008 field contained anything other than "eng". There were 12,350 "non-eng" records (13% of the file). Excluding these reduced the size of the dictionary from 47,040 to 30,810.

6.4.2 Selection of candidate replacements

Examination of Okapi '84 searches had shown that around two-thirds of misspellings retained the approximate consonant structure, and it is well documented that initial letters are rarely wrong [9, 10].

We decided to use a soundex-type procedure in which the first letter of each word is left unchanged, vowels and a few consonants ("h" and "w") are ignored, doubled consonants made single and the remaining consonants divided into approximate phonetic groups. If adjacent consonants are in the same group only a single code is output. Some information is retained from the vowel structure: if two consonants are in the same phonetic group but are separated by one or more vowels (or "h" etc) the code for that consonant group is output twice. Also, a vowel other than "e" at the end of a word is represented by a "vowel code".

The algorithm is given in Appendix 1.

We also tried a similar procedure which retained more of the vowel structure by representing any vowel or sequence of vowels by a "vowel code". This gave shorter candidate replacement lists, but sometimes missed easy corrections (e.g. HOIRZONS --> HORIZONS).

Examples:

```
economics )
economic  ) --> ecmmc  (repeated 'm' because of intervening
*ecomonic )             vowel)

*econmic    --> ecmc

rabbit )
rabid  )
rapid  )  --> rbd
repeat )
(and many others)

sociology  )
*socialogy ) --> sclcy
*sociolgy  )
```

The access keys for the dictionary are Soundex codes. Each Soundex code has a pointer to the list of words which give rise to it. That is, it is constructed as an inversion of the examples above - the arrows point the other way:

```
ecmmc --> (economic, economics, .. )
...
rbd   --> (rabbit, rabid, rapid, repeat, .. )
...
sclcy --> (sociology, .. )
```

6.4.3 *Finding the nearest match*

When a word in a user's search statement does not stem to anything in the index, the word is encoded using the same algorithm (given above) which was used to construct the dictionary. The code is looked up in the dictionary. If it is not found, the procedure terminates. If it is found, the code's associated word list is scanned sequentially. Each word in the list is compared with the user's original word, and a "matching score" calculated. If there is any word with a high enough score, the one with the best score is offered to the user. If there is more than one with the same score, the first is offered (effectively arbitrary).

```
CAN'T FIND 'sociolgy' - closest match found is 'sociology'

GREEN KEY  to use 'sociology' instead
BLUE KEY   to type a different word
```

Any word in the candidate list has the same initial letter as the user's word and a similar consonant structure, so we guessed that it might be unnecessary to take any further account of the order of the letters. We use a simple "anagram" technique and a word-length criterion. The minimum acceptable score is relatively higher for rather short words than it is for longer ones.

The matching algorithm is given in Appendix 2.

Example:

```
User's word  *APLIANCE  encodes to  ABLMC

Lookup of  ABLMC  returns a list consisting of

AFFLUENCE  which scores 4 (4 letters apart from the first
                           letter in common with user's word)
APPEALING  which scores 6
APPLIANCE  which scores 7
APPLYING   which scores 4

APPLIANCE has the highest score and length within one letter of
the user's word, so this is suggested as a replacement.
```

(Note that APPEALING would have been offered if APPLIANCE had not been in the dictionary. This illustrates the importance of having a dictionary of adequate size.)

6.4.4 Discussion of the spelling correction technique

USER INTERACTION

See Chapter 7, Figs 7.8 and 7.6 for the screen layouts.

Words whose stems are not found in the catalogue's index fall into (at least) three categories. They may be correct or incorrect, and, if incorrect, the corrected word may or may not be in the catalogue.

Hence the procedure is not offered to the user as spelling correction, but rather as the neutral 'CAN'T FIND "‹word›" - closest match found is "‹suggestion›"'. It would sometimes be presumptuous to say 'CAN'T FIND "‹word›" - do you mean "‹suggestion›"' and out of the question (with our methods) ever to make an *automatic* substitution.

One of our objects was to avoid having to process searches which contain any words which are not found. Ignoring the word can result in the retrieval of rubbish. Implicit AND systems will simply return a failed search. In ranked output systems such as ours it is impossible to know what importance to attach to a "missing" word. Anyway, a majority of such words are misspellings or miskeyings. It is essential that the user should know that a word is not found. Hence the system forces the user either to replace a missing word or to tell the computer to ignore it.

Why not offer more than one word?

If more than one word from the candidate list scores highly in the matching procedure it would seem desirable that they should all be offered. In an earlier version of Okapi we experimented with a similar procedure for personal names (which is what the original Soundex scheme was intended for). That version would offer up to nine possible matches arrayed neatly on the screen for selection by keying a single digit. However, we assume that whereas a user may genuinely not know how to spell a personal name, a large proportion of erroneous words are due to miskeyings rather than misconceptions about the spelling. Indeed, Okapi '84 logs show that personal names are more likely to be right than other words. A small sample of erroneous words from subject searches suggests that about 90% look like miskeyings rather than misspellings.

Simplicity (for the user) outweighed other considerations, so we decided only to offer at most one suggestion for replacement. Where more than one dictionary entry gives a high matching score, it would be sensible to offer the most frequent (as being *a priori* more likely to be searched for), but we did not have time to implement this.

Further, there is usually a single clear winner, and it is nasty *computerese* to offer a "choice" from a set of size one. The alternative (selection by digits if more than one, our existing layout if only one) is worth trying, but might easily "throw" a user who has become used to using the green key to select the computer's suggestion and is now faced for the first time with the multiple choice screen.

We carefully monitored all occurrences of the automatic correction during the first week of live use, before data collection had started. In an appreciable proportion of occurrences a user sat for a long time staring at the screen and/or pressed the red or black keys to abort the search even though the suggested correction was good. This reinforced our feeling that it would be unwise to offer a choice of corrections.

6.5 Search processing and term combination

Between the user's entry of a search and the system's display of the result, the following steps are carried out:

1 User's input is preprocessed. System displays

> Your search 'electrical safety standards'
>
> Looking up these words

2 Lookup of weak and strong stems in the index, referring back to user if any not found. System displays each word with the number of books (if any) indexed under the word's weak stem. Several other messages are possible, the most frequent being CAN'T FIND '<word>'

When everything has been looked up, including strong stems if this is the *EXP* catalogue, the system decides which of the items are going to be used in the merge (below).

3 Assignment of weights to terms.

4 Calculation of "good" and "acceptable" weights for record retrieval.

5 "Merging" of the posting lists for the terms to find records which reach an acceptable weight.

6.5.1 Preprocessing and index lookup

The user's search statement is disassembled into words, each word is weak-stemmed, and the statement reassembled. Each component is looked up in the weak stem index unless it is in the stop list. What constitutes a *component* is actually *determined by* the index lookup - in the *EXP* system

this may be a word stem or a phrase from the *go/see* list; in *CTL* it is always a stem; in *OSTEM* it is always a "raw" word. Any component which is not found, or which is found but has no postings (the latter case can only occur in *EXP* with the few *go/see* terms which do not occur in the source file) is negotiated with the user, who must replace it, tell the computer to ignore it, or terminate the search.

In the *EXP* system each weak stem which is not in the *go/see* list is also strong-stemmed, and the strong stem is looked up in the index. Each strong stem is marked as being semantically equivalent to its corresponding weak stem.

At this stage the system has a list of components with a (non-zero) number of postings for each component. It knows whether an item is a strong or a weak stem, and, if it is a strong stem, which if any of the weak stems it is equivalent to. No component is included more than once, and no account is taken of word order - except in the case of phrases in the *go/see* list.

Example:

ELECTRICAL SAFETY STANDARDS FOR ELECTRIC FIRES

produces the weak stems

1	ELECTRICAL	(573 postings)
2	SAFETI	(262 postings)
3	STANDARD(1)	(565 postings)
4	ELECTRIC	(421 postings)
5	FIRE	(141 postings)

and the strong stems

6	ELECTR	(888 postings)	equivalent to 1 and 4
7	SAFETI	(262 postings)	equivalent to 2
8	STANDARD(1)	(579 postings)	equivalent to 3
9	FIRE	(141 postings)	equivalent to 5

(1) The weak stem STANDARD arises from STANDARD and STANDARDS, but the strong stem also includes STANDARDISATION.

The purpose of the equivalences is to prevent a concept contributing twice to the weight of a retrieved record. We don't want the search in the example to retrieve a record MARINE SAFETY STANDARDS AND STANDARDIZATION, as it probably would if both the strong and weak stems "STANDARD" contributed to its weight. On the other hand, if the user has entered two morphologically similar words (ELECTRICAL and ELECTRIC on the example) we do not count these as equivalent although they have the same strong (but not weak) stems. We did give some consideration to marking any two

terms with the same strong stem as equivalent, but decided that this would introduce too much fuzziness: in the search COMMUNICATION IN COMMUNIST SOCIETIES, COMMUNICATION and COMMUNISM would be treated as equivalent because their strong stems are the same. Items indexed under all three terms would not be ranked any higher than items indexed under only two.

6.5.2 Assignment of term weights

Each component is given a weight which is the largest integer not greater than log(N/n), where **n** is the number of postings for the component. **N** is a constant which is related to the number of records in the bibliographic file. It must be at least as great as the number of postings for the commonest term in the index. The logarithm is taken to base 2 (theoretically it doesn't matter what base is used, provided the weights are stored with enough precision, but since we store them as one-byte integers, base 2 gives a reasonable spread of weights with minimal arithmetic).

Example:

For the systems used in this project with the current PCL catalogue the weight constant **N** is set to 32768 (this being comfortably more than the number of postings for the commonest term in the index). Since $2^{15} = 32768$, log(32768) is 15, and the weight of a term with n postings is 15 - log(n) (rounded down if necessary).

Thus the weights are

	term	pstgs	weight
1	ELECTRICAL	573	15 - 9 = 6
2	SAFETI	262	15 - 8 = 7
3	STANDARD	565	6
4	ELECTRIC	421	7
5	FIRE	141	8

and for the strong stems

6	ELECTR	888	**6**
7	SAFETI	262	**7**
8	STANDARD	579	**6**
9	FIRE	141	**8**

For a theoretical basis for this weighting scheme, see [11]. It is said to give the best probabilistic approach to ranked output in the absence of any relevance information (or in the absence of any knowledge about the relative importance of the terms). A term which occurred in every record would be useless for discriminating between relevant and non-relevant records, so this should have zero

or very low weight. Our weight function satisfies this criterion (if **N** is set to the number of records in the file). It is not always true that rare terms are more important than common ones, but at least if records containing rare terms are presented first it doesn't take too long to look at them and get them out of the way.

An example of a type of search where the rarest term is by no means the most important is LEAST SQUARES ESTIMATORS. "Estimators" has only two postings in the PCL file, although its strong stem has several hundred. The problem with this search is that it comprises a single concept, and so should be treated as a phrase, but the same concept could be, and is, expressed in relevant records as LEAST SQUARES METHOD or just LEAST SQUARES.

6.5.3 Calculation of "good" and "acceptable" weights for record retrieval

During the merge (see below), the weight of a record is determined by the weights of the terms which it has in common with the search.

The *minimum acceptable weight* (*MAW*) is the threshold weight for a record, below which it will not be retrieved (although it must be indexed under at least one of the terms of the search, otherwise it would not be considered at all).

The *minimum good weight* (*MGW*) is the least weight at which a record will be considered as a reasonable match with the user's search.

MAW and MGW are calculated using the *maximum possible weight* (*MPW*). MPW is the weight which a record would have if it contained all the terms of the search. It is the sum of the weights of all the weak stems in the search. In the example above these add up to 33.

The method used to calculate MAW and MGW depends on the number of terms in the search. Searches are treated differently depending on whether they contain one term, two terms or more than two terms. When there are only one or two terms the actual function used for the weighting is almost irrelevant, provided that strong stems have lower weight than weak stems.

SINGLE TERM SEARCHES

For *CTL*, MAW = MGW = MPW = the weight of the (single) weak stem. For *EXP*, MGW as *CTL*; *MAW* = weight of the strong stem. A large proportion of single term searches are for proper names, where the strong and weak stems are generally identical.

Example:

INTEGRALS (*EXP* system) retrieves 46 books under INTEGRAL(S), followed, if the user wishes, by another 230 under INTEGRATION, INTEGRATING, INTEGRATED and other words which give the strong stem INTEGR.

TWO-TERM SEARCHES

This is the most frequent number of terms in a search. In the absence of semantic knowledge, it is only the notions "common" and "rare", together with the ability to differentiate between strong and weak stems, which are needed to rank the records for output.

TWO COMMON TERMS

Only records containing *both* terms (or their strong stems) will be retrieved.

MAW = sum of the weights of the strong stems. MGW = sum of the weights of the weak stems. (In the *CTL*, MGW = MAW.)

Example:

INDUSTRIAL SOCIETY

ONE COMMON TERM, ONE RARE

All records containing (the strong stem of) the rare term are retrieved.

MAW = weight of strong stem (*EXP*) or weak stem (*CTL*) of the rare term. MGW = sum of weights of strong stems (*EXP*) or weak stems (*CTL*). All records which contain the rare term are retrieved.

Example:

HISTORY OF SWORDS

TWO RARE TERMS

All records containing (the strong stem of) either term are retrieved.

MAW = weight of strong (*resp* weak) stem of commoner term.
MGW = sum of weights of strong (*resp* weak) stems.

Example:

YACHTING AND BOATING

MORE THAN TWO TERMS

Records will usually be retrieved if they contain (stems of) about two-thirds of the terms in the search.

MAW = half the maximum possible weight, MGW = two-thirds of the maximum possible weight.

Example:

SOCIAL STRATIFICATION AND OCCUPATIONS

The frequencies and weights (strong stem weights not given) are

	term	pstgs	weight
1	SOCIAL	6257	15 - 12 = 3
2	STRATIFICATION	46	15 - 5 = 10
3	OCCUPATIONS	100	15 - 6 = 9

MPW = 3 + 10 + 9 = 22
MAW = 11 (half of 22)
MGW = 14 (two-thirds of 22)

Thus all records containing any two of the three terms will be retrieved. The strong stem weights are not given here, but in this example the retrieved set would probably include records indexed under any two of the strong stems alone.

In the PCL file there are two records indexed under all three words, 41 more under SOCIAL and STRATIFICATION, and a further 13 under SOCIAL and OCCUPATIONS. (Both the records under STRATIFICATION and OCCUPATIONS also contain SOCIAL.)

When the merge (see below) is complete the user will see

> 2 books match your search exactly
> (56 books found altogether)

6.5.4 *Merging the posting lists*

The postings lists in the index are ordered. They can be thought of as being in document number order. In the present implementation "document numbers" are really disk addresses. Whatever they are they must be ordered, otherwise the merge would be too inefficient, and they must be able to tell the system where to fetch the records from for display to the user.

Up to 16 postings lists are merged into a single output list. A list represents a single weak or string stem. The lists are numbered 1 to 16. Each list has a weight attached to it, and possibly a list of the numbers of other

lists to which it is semantically equivalent.

While there is some list which is not finished, the merge finds the "smallest" posting not yet considered. It sums the weights of each list in which this posting occurs, omitting weights for those lists which are marked as equivalent to another list which contains this posting. If the total weight for the posting is at least MAW (minimum acceptable weight), the posting is copied to the output list. Thus the merge ends with a list containing the addresses of all the records which contain enough of the components of the search to reach MAW.

> In our implementation the output list is restricted to 512 postings, but all postings in the input lists are considered: if the output list becomes full new postings replace postings already in the output list if their weight exceeds that of the posting with lowest weight in the existing output list. This makes the process more complicated than it need be, but we were working with computers with a very limited amount of core memory. It is not a constraint to the user: 512 records is comfortably above what most users wish to see, and the list is guaranteed to contain the "best" records.

As soon as the merge is finished, the output list is sorted by weight so that the records with the highest weight will appear first.

6.6 The bibliographic file

This is very similar to the one used for Okapi '84, which is fully described in Chapter 4 and Appendixes 1 to 4 of the first Okapi report [12].

To try to add a bit more subject information the present file includes additionally MARC 651, and subfields $x (subject or form subdivision) and $z (place subdivision) of 650 and 651. We also intended to use 505 (contents notes), which had only been used for records with analytical entries in the previous file, but this is so rarely used in the PCL file that it was not worth the overhead of an empty field in nearly every record.

Nevertheless, by British standards the PCL records are comparatively rich in subject content as a large proportion of them contain verbal feature headings (MARC 083), PRECIS headings and LCSH. Inevitably, there is a good deal of duplication of headings, but most records look reasonable when displayed.

The extra subject information increased the average record length considerably. To compensate for this we no longer store copy numbers, but only a count of the number of copies at each site.

There are about 98,000 records in the file.

6.7 The subject index

Index structure and storage are very similar to that described in 5.7 of [12]. Since there are only four data types in the index (weak stems, strong stems, entire and truncated Dewey numbers) the structure has been simplified a little.

6.7.1 Indexing and the go/see list

The *go/see* list is used during indexing. When index terms are being extracted from bibliographic records the field being indexed is matched against the list before the normal process of word extraction is performed. If it contains a phrase or word from the list a token representing this *go/see* entry is output. For example, "United States" produces the same token as "USA". When the final index is being produced the actual entries (e.g. "United States") are read in at the right place in the alphabetic sequence, and pointed at the list of postings for the appropriate token. The end result is that searches for "United States" and for "USA" return the same posting list. Thus the *go/see* list is no longer explicitly used after the file has been indexed; it is, in effect, incorporated in the index. (The system does not "know" at search time *which* terms are included in an equivalence class. But it does know when it has found something which is in the *go/see* list. It can only inform the user that "UNITED STATES" includes "USA" unless both these terms occur in the same search (7.4.2).

6.7.2 Source fields

The index is generated from

all title-like fields

subject headings and verbal feature headings

corporate and conference names (both author and subject)

the *go/see* list.

6.7.3 Index contents and size

The index used by *EXP* and *CTL* contains weak and strong stems of every word in the source fields.

Hyphenated words contribute both "concatenated pairs" and their separate constituents ("non-proliferation" gives rise to "non" and "proliferation" and "nonproliferation", as well as their strong stems "prolif" and "nonprolif").

"Initialisms" are processed so that they become words ("USA" = "U.S.A." = "U S A").

Every entry from the *go/see* list, weak stemmed, is in the index (6.7.1).

The index also contains Dewey numbers and truncated Dewey numbers, but these were not used in the experiments described in this report.

Without Dewey numbers, the mean number of index terms per bibliographic record is about 24. There are about 61,500 distinct stems. For the majority of words, the strong stem is the same as the weak stem. When this is the case they are not stored separately, but merely flagged as being both strong and weak stems.

6.8 Storage requirements

Bibliographic file: 20 megabytes

Stem index (used by *EXP* and *CTL*): 5.7 megabytes (excluding Dewey numbers, which occupy about another megabyte)

Word index (used by *OSTEM*): 4.4 megabytes

Spelling dictionary and its soundex code index: 0.6 megabytes

6.9 The Okapi '86 programs

It is often true that the more simple a system appears to the user, the more complex it needs to be "behind the screen". We would not have been able to undertake this research if we had not had the Okapi '84 system to build on. Several person-years of design and programming work had gone into this. This meant that, for this project, we did not have to spend much time on file structure and index lookup, or on record displays. The programs which deal with the formatting and sequencing of record displays consist of some 2000 lines of code; this is largely unchanged from the way Gill Venner wrote it three years ago. However, a very considerable amount of new design and programming had to be done.

Like previous Okapi systems most of the programs are written in Z80 assembly language. The programs for reading MARC tapes and for selecting and stripping the records are written in COBOL and DEC assembly language.

There seems to be a tendency for the complexity of programs to vary inversely with the outward simplicity and openness of the system. Why this is so is illustrated by the trivial example of a program which allows a user to enter dates in free formats, such as 1/3/87, 1st March 1987,

1.iii.87. Such a program is about an order of magnitude more complicated than one which rejects a date if it is not entered in some "standard" form like 01-03-1987. Until the advent of computing for the general public it was not usually worth while to write programs like this.

It would be out of place here to give a detailed desription of the internal workings of Okapi '86, but it does suffer from the type of interactional complexity illustrated above.

An example is the program which takes control while the user's search is being parsed and its terms looked up. This program has to handle the various combinations of messages which can appear on the "searching" screen (Fig 7.5 etc) while it is working, and maintains links between what the user typed and the stems which are being looked up. It appears very simple, yet it contains about 1700 lines of code at the top level (and several times as much again at lower levels which deal with index lookup etc). It is controlled by three decision tables, the largest of which has to check five conditions (is this a phrase or a word, did it come from the *go/see* list, has it any postings, have we had it before in this search, and, if it has occurred before, did it arise from the same word or words in the user's search statement?); depending on which of the conditions are true is has to perform various combinations of seven actions (store the result of this term-search, display "N books under ...", display "... included under ... ", perform strong stemming etc). For each term in the search there are, at the top level, about 40 different paths which the program can follow. Even with this degree of complexity there are "loose ends" (surprisingly, we have not come across any actual mistakes); a few rare combinations of conditions are not properly dealt with.

References

1 ULMSCHNEIDER J E and DOSZKOCS T E. A practical stemming algorithm for online search assistance. *Online Review 7* (4), August 1983, 301-315.

2 MISCHO W. Library of Congress Subject Headings: a review of the problems, and prospects for improved subject access. *Cataloging & Classification Quarterly 1* (2/3), 1982, 105-124.

3 SCHABAS A H. Postcoordinate retrieval : a comparison of two indexing languages. *Journal of the American Society for Information Science 33* (1), 1982, 32-37.

4 MITEV N N and WALKER S. Intelligent retrieval aids in an online public access catalogue : automatic intelligent search sequencing. In: *Informatics 8: Advances in intelligent retrieval. Proceedings of an Aslib/BCS conference. Oxford, 16-17 April 1985.* London : Aslib, 1985.

5 PAICE C D. *Information Retrieval and the Computer.* London : MacDonald and Jane's, 1977.

6 PORTER M F. An algorithm for suffix stripping. *Program* 14 (3), 1980, 130-137.

7 LENNON M. *and others.* An evaluation of some conflation algorithms for Information Retrieval. *Journal of Information Science* 3, 1981, 177-183.

8 FRAKES W B. Term conflation for information retrieval. In: *Research and development in Information Retrieval : proceedings of the third joint BCS and ACM symposium King's College, Cambridge, 2-6 July 1984.* Edited by C J van Rijsbergen. Cambridge University Press *on behalf of* the British Computer Society, 1985, 383-389.

9 TAGLIACOZZO R, KOCHEN M and ROSENBERG L. Orthographic error patterns of author names in catalog searches. *Journal of Library Automation* 3 (2), June 1970, 93-101.

10 POLLOCK J J and ZAMORA A. Collection and characterization of spelling errors in scientific and scholarly text. *Journal of the American Society for Information Science* 34 (1), January 1983, 51-58.

11 CROFT W B and HARPER D J. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation* 35 (4), Dec 1979, 285-295.

12 MITEV N N, VENNER G M and WALKER S. *Designing an online public access catalogue : Okapi, a catalogue on a local area network.* (Library and Information Research Report 39). London : British Library, 1985.