3 Stemming and truncation

3.1 Introduction

One way of broadening searches in information retrieval is to use systematic abbreviation of words so as to bring together words which are morphologically related, in the hope that they will also be semantically related. This can be done manually or automatically.

Information retrieval intermediaries use manual truncation to conflate words which are both morphologically and semantically related. Intermediaries use their linguistic knowledge to avoid drawing in words which might seriously decrease the precision of the search. Truncation is often combined with boolean OR to bring in other synonyms. For example, the concept "communism" might be submitted to an IR system as "communis* OR marxis*" rather than as "commun* OR marx*" which would lead to the retrieval of records indexed under "communication" etc.

Manual truncation is not a particularly easy or natural skill to acquire, and cannot be considered for casual catalogue users. It is a facility which can be provided for those, such as library staff, who wish to learn how to use it. Many of the "keyword" type online catalogue systems do allow for manual truncation [1].

The discussion here is limited to processes of *automatic* abbreviation or truncation which aim to conflate related words by reducing them to their *stems*. The word-segments to be removed are referred to as *affixes*. An affix can be a suffix (like "ation") or a prefix (like "pre"). Many prefixes cannot safely be removed except in narrowly defined subject areas (such as chemical terminology), because they tend to have a more drastic effect on the meaning of a word than do suffixes, which are often inflexional.

The next three sections form a fairly condensed survey of some of the stemming techniques which have been published. They are of a rather technical nature. Readers who are primarily interested in online catalogues might prefer to skip to Section 3.5.

3.2 Methods and techniques in algorithm construction

Many algorithms have been reported. Some of the ways in which they differ are outlined in the following sections.

3.2.1 Iterative or longest match?

An algorithm can be iterative in method, it can use a "longest match" method or a mixture of both.

The iterative method removes suffixes one by one from the stem (sometimes single letters or groups of letters are removed rather than true suffixes, but the same principal of gradual reduction still applies). For example, Porter's iterative algorithm [2] processes the word "conformability" in three iterations: at the first pass, "y" would be replaced by "i"; at the second iteration, "biliti" would become "ble"; finally "able" would be removed, leaving the stem "conform".

The longest match method removes the longest matching affix in one iteration. In the above example, the longest suffix ("ability") would be removed in one step.

3.2.2 Conditional rules

These include rules which prevent the removal of a given affix or class of affixes if a given condition is not satisfied. They commonly involve a minimum length condition: "remove terminal 's' unless the result would be less than four characters long" is an example. Some stemming procedures have word-specific rules like: "remove 'ed' unless word is 'united'".

3.2.3 Stem modification

Particularly in iterative procedures, better conflation can be achieved by rewriting some stems. For example, terminal "y" may be replaced by "i" to conflate word forms ending in "y" with other related words: this rule would apply to the words "pony" (to retrieve "ponies" when the "es" suffix is removed) or "happy" (to retrieve "happiness" when the "ness" suffix is removed). Another example is the changing of "pt" to "b" to conflate word forms ending in "b" with grammatically related words which change the "b" to "pt": this rule would apply to the word "absorption" (to retrieve "absorb" and "absorbent").

3.2.4 Compilation of the suffix list

The differences above correspond to differences in the structure of the algorithm. All algorithms, whether iterative or longest match, need a list or dictionary of suffixes. Dictionaries can be constructed manually, or they can be generated automatically or semi-automatically from bodies of text. Their size and scope will strongly influence the behaviour of the algorithm.

Lennon and his colleagues [3] suggest that while manual evaluation of lists of possible suffixes gives results of a very high quality, the length of time taken often makes this method impractical. In their evaluation, they test a method (using the frequency of word endings) for the automatic generation of possible suffixes. They concluded that "fully automated methods perform as well as procedures which involve a large degree of manual involvement in their development" (183).

3.2.5 Users' needs

The main function of conflation algorithms must be to improve recall; there will always be some searches where there is a loss of precision. The balance between recall and precision must be chosen to suit different classes of users. An industrial user of a retrieval system who needs a comprehensive search might be prepared to examine a substantial proportion of irrelevant material (caused by overstemming). For general library catalogue use, on the other hand, under-stemming is to be preferred to over-stemming.

3.3 Conflation algorithms: a review

3.3.1 INTREX

One of the first conflation algorithms to be developed and tested was part of the Project INTREX (see Overhage and Reintjes [4] for a general review). Lovins [5], who participated in this project, produced a list of suffixes by first examining a preliminary list generated from the endings of words from the Project INTREX catalogue. The list was used to see when the use of a given ending from the word in the dictionary would result in a mismatch, or in the omission of a stem which ought to match. This manual assessment allowed the author to refine the list of endings and to compile word specification and recoding rules. The final list contained around 260 suffixes. It was used in conjunction with both context rules and recoding rules.

3.3.2 RADCOL

Lowe and others [6] tested two algorithms as part of the RADCOL project. The first used two passes through a list of 95 suffixes; the second used a single pass longest match algorithm with a longer list of 570 suffixes. After tests, the second algorithm was adopted. Lowe and his colleagues obtained this list by a multi-stage process. First the characters of the most frequent words in the index were reversed, and the reversed words were sorted in alphabetical order. Then a minimum string length was established. The list was scanned for repeated character strings, on the assumption that strings which occurred with more than a certain frequency might be possible suffixes. Finally, these character strings were examined manually and suffixes were selected from them. The comprehensiveness of this suffix list meant that the number of context and recoding rules could be reduced, increasing the simplicity of the algorithm.

3.3.3 Generation of suffix lists

Lennon and his colleagues, in the course of their evaluation of conflation algorithms [3], extended the method used in the RADCOL Project and in the INSPEC project (discussed below). A list of reversed words was used to produce a list of word endings which occurred with more than a certain frequency. These can be assumed to be suffixes although if this algorithm is to be used automatically, no recoding is possible. This is a simple but unwieldy approach. Since it operates as a longest match algorithm, the inclusion of a string "iveness" in a suffix dictionary also necessitates the inclusion of the substrings "veness", "eness", "ness" and so on. The proportion of strings which have a real utility is therefore reduced.

This method was also used by Tarry [7] to generate several sets of equifrequent character strings from the ends of words. The method involves selecting from a body of text character strings of variable length occurring with approximately equal frequencies and with low sequential dependence. This suffix generation procedure can be used for automatically determining *subject-specific* or *language-specific* lists of suffixes. The incentive for using this technique for suffix generation was the supposition that character strings representing suffixes would occur more frequently than other terminal character strings. As well as this, it has been observed that letter dependency within words decreases at the boundaries of word units such as affixes. Tarry's algorithm works on the longest match principle and has no restriction on suffix removal other than that the remaining stem should be of a minimum length. Since there is no restriction on removal the algorithm is context-free and uses neither recoding nor partial matching. Tarry justifies this approach by the desirability of eliminating "the large amount of manual preprocessing required, both in the construction of the suffix lists, and in the formulation of the suffix removal rules" [7, p21]. This algorithm was compared with the INSPEC algorithm; retrieval tests with the Cranfield 1400 test collection were made and it was found that the algorithm performed at least as well as the traditional algorithm [7, p78].

3.3.4 INSPEC

A conflation algorithm was designed by Field [8] at INSPEC with British Library funding. The list of suffixes was compiled manually after consulting a Key Letter In Context (KLIC) index. This algorithm uses a mixture of longest match and iterative suffix removal and incorporates several features which were designed to improve its effectiveness: minimum stem length, recoding rules and three stage conflation. This last application is particularly interesting. The word to be conflated is first dealt with by Algorithm 0 which removes stop words and common endings such as plural forms (this stage is partially iterative). Words which are not stopped are then treated by Algorithm 1 which removes all other suffixes which are present in a longest match routine. In a final stage, Algorithm 2 makes adjustments to the stem, usually on the basis of stem length. Field claims that this use of a three stage process increases the overall efficiency.

3.3.5 Stemming in SMART and FIRST

The IR systems used in the SMART projects incorporated stemming. The SMART system bases all dictionaries on word stems rather than original words. The suffixes which generate the word stems are listed in a suffix dictionary, and each one carries one or more syntactic codes. These must be matched with complementing codes attached to the word stems in order to determine which suffixes match which stems [9, p32].

Dattola [10] has described FIRST - the Flexible Information Retrieval System for Text - which is based on the methods developed during the SMART project. The most important part of this procedure is a stem dictionary; this is the basis of the conflation procedure. Words are added to the stem dictionary if they fail to match an existing stem and are more than three characters long. This method uses a stem dictionary of *whole* words rather than actual stems; new words are not added to the stem dictionary if they are suffix variations of existing stem entries.

3.3.6 MORPHS

Bell and Jones have described the retrieval system MORPHS in a number of articles including [11]. This system (the name means "Minicomputer Operated Retrieval (Partially Heuristic) System") is used at the Malaysian Rubber Producers' Research Association. It incorporates automatic stemming. Bell and Jones [12] discussed the use of *roles* and stemming in an earlier version of the system as a means of improving recall and also incorporating some syntactic knowledge. They believed that the two techniques could be combined by replacing the suffixes by a limited number of role indicators. In this system stemming was performed manually by the searcher who could either, as in their example, search for MIX; or MIX (role A) - to include MIXING; or MIX (role D) - to include MIXED. An extensive suffix list is used; its size is increased by its treatment of exceptions (the stems "cation" and "station" are included and are used in preference to the stem "ion") and by the inclusion of chemical suffixes. The system attempts to guard against the removal of apparent affix strings ("pre" in "pressure" for example) by checking that the stem is present in the stem file before the affix is removed. A minimum stem length also helps to protect against the removal of apparent affixes. Bell and Jones recount how they were puzzled by the generation of the stem "im" until they discovered that the prefix "an" and the suffix "al" had been stripped from the word "animal". Both longest match and iterative methods are used in the removal of suffixes and the creation of stem dictionaries.

3.3.7 Cercone and linguistic analysis

Some algorithms have been developed and tested for natural language applications. The morphological algorithm of Cercone [13] aims to determine the root of a term by removing suffixes and prefixes. Affixes are removed iteratively by consulting an affix dictionary. This process uses a system of order classes, assuming that affixes are added to the stem in a certain order (this particularly applies to suffixes). By using these order classes a word which is to be stemmed can be examined for the presence of affixes belonging to each class in turn. The removal of some affixes is followed by recoding of the root. After recoding, the root dictionary is searched and, if a match is made, the root and the affixes are output. If there is no match the next order class of affixes is scanned and the process continues until a root is found. Cercone's algorithm was designed for use in textual analysis and relies on the manual construction of lists and rules. It is discussed by Cercone [14] as an element of morphological analysis and lexicon design for natural language processing.

3.3.8 MARS

A different approach has been taken by research workers at Siemens who have developed a system called MARS [15]. This system uses a morpheme lexicon to decompose words rather than a affix-stripping algorithm. This makes it possible to incorporate linguistic analysis rather than morphological matching alone. The authors feel that "retrieval operations like left truncation, right truncation, and masking are noticeably inadequate as regards their ability to filter out inappropriate terms and expand upon more useful ones" [15]. In their discussion the authors compare their approach exclusively with truncation saying that truncation is insufficiently powerful in recall and precision. They do not include conflation techniques in their review. As the review above has demonstrated, the incorporation of recoding rules into a conflation algorithm can substantially improve precision. The operation of MARS is described below; it should be borne in mind that any potential improvement in precision and recall should be balanced against the operational costs and computer storage required.

MARS has morpheme dictionaries and grammar rules for each language. They are used to split words into prefix, stem, derivational and inflectional elements. The extracted word stems are collected in a stem-file in which pointers back to the textwords containing the particular stem can be followed, enabling retrieval of these words. The morpheme dictionary contains affixes, inflectional endings and fillers. Each entry is stored with a 32-bit string indicating special morpheme characteristics and certain compositional properties. The morphemes in the dictionary are the longest possible strings obtainable from all of the possible derivations ("traditionality" for example would be viewed as a derivation of "tradition" and not "trad(e)"). This morpheme dictionary is supplemented by two smaller lists. One includes "irregular" stems like Latin and Greek plurals and irregular verb forms. The other list contains strings which regularly undergo graphemic change (like "y" to "ie"); these transformations are processed automatically. A pre-processor checks to see if string transformations are necessary. After this, the three lists are used by a decomposition grammar which deals with each word. After having reached a certain stage in a word (a prefix for example) certain conditions have to be fulfilled if the word is to be passed to the next stage. These conditions are listed in the morpheme grammar for the language.

MARS was tested by a retrieval expert who carried out twelve real searches, once with and once without MARS. Recall was increased by 68% when MARS was used. Moreover, this was achieved without a significant decrease in precision (this did decrease but only by 7% from 68% to 61%). There were difficulties with compound words and phrases and with verbs; these were caused by limitations within the structure of MARS and can be offset by modifications to the search strategy used.

3.3.9 Porter

An iterative algorithm was developed by Martin Porter [2] at the University of Cambridge Computer Laboratory. He uses a concept which he calls the "measure" of a word. This is the number of vowel-consonant transitions in the word. It is used in some of the conditional rules: for example "remove terminal 'ance' if the measure is greater than one". The algorithm is a five-step, partially iterative procedure using a dictionary of around 60 suffixes. Porter notes that a point is reached in the development of a conflation algorithm when the inclusion of additional rules to improve performance in one area leads to a corresponding decrease in performance in another area. He warns that unless this tendency is guarded against it is very easy for the algorithm to become more complicated than it need be. There is a temptation to try to deal with word-forms which appear to be important but which are rare in most applications. He cites the examples "deceive/deception", "resume/resumption" which occur very infrequently in the vocabulary of most indexes. Since there will always be some error rate, Porter argues that it is not worthwhile trying to cope with these cases.

Porter's algorithm is simple. It has few rules and a small dictionary, and so is economical of computing time and of storage. It was tested and found to achieve a comparable (actually slightly better) level of effectiveness than the algorithm [16] used previously at the Cambridge Computer Laboratory (see below for discussion).

Porter's algorithm is the one used by Frakes [17] in the CATALOG retrieval system. This incorporates a default naive user mode which is based on the Paperchase system (2.4.3). An experienced user can override the defaults and use an alternative command mode. Moreover, whereas Paperchase only included truncation, CATALOG incorporates an automatic conflation algorithm. A user can do manual truncation, but the default is to search for related terms automatically. Related terms are presented to the user, with the number of occurrences. The user can select from this list or indicate that all terms are to be used. Frakes has written that the CATALOG retrieval system has been shown to be "feasible" [17] but this conclusion is not based on an extensive test of Porter's algorithm in a real environment.

3.3.10 Dawson

Dawson's algorithm [16], referred to above, was based on that developed by Lovins [5], but extends Lovins' initial list of about 250 suffixes sixfold to about 1,200. It was anticipated that the size of this suffix list might create problems of storage and processing time. Dawson coped with this large suffix list by reversing the suffixes (and word specific suffix removal conditions) and indexing them by length and by final letter. This algorithm used a longest match method. Unlike most of the algorithms, Dawson does not use recoding; instead, there are classes of stem ending, and if two stems match up to a certain number of characters and the remaining characters of each stem belong to the same stem ending class, then the two stems are conflated to the same form. Dawson includes fifty of these stem ending classes. An example of this might be with the stems "absorb" and "absorpt"; by including "-rpt" and "-rb" in the same stem ending class these two stems can be conflated to the same stem.

3.4 Evaluating conflation algorithms

The effectiveness of stemming algorithms can be evaluated by assessing the degree to which terms are overstemmed and understemmed. One measure is the proportionate decrease in the number of distinct terms after stemming.

Lennon and others tested five conflation algorithms [3] as part of an evaluative study. They confirmed a previous suggestion by Landauer and Mah 1980 [18] that the the RADCOL algorithm tended to overstem (reducing "posed", "positively" and "positioning" to "pos"). The Porter algorithm tended to understem (reducing "accuracy" to "accurac" but "accurate" and "accurately" to "accur").

These conclusions are supported by the compression results which were achieved by Lennon and his colleagues. With the Brown Corpus, Porter achieved the least compression (38.8%) and RADCOL achieved the greatest (49.1%). The other algorithms tested achieved 45.5% (Lovins) and 47.5% (INSPEC). Several test databases were used and while the percentage compression achieved did vary significantly according to database, the relative compression achieved by different algorithms was similar. Retrieval tests demonstrated that algorithms which tended to stem generously did not necessarily increase retrieval effectiveness; the Porter algorithm tended to understem, but it performed better in the test than the RADCOL algorithm which tended to overstem. The INSPEC algorithm, on the other hand, is also a strong algorithm, but this gave the best precision orientated search. Lennon and his colleagues also performed a test for recall effectiveness. In this test, a similarity measure using trigrams performed well; but the Porter algorithm performed as effectively. They conclude that "... there is no relationship between the strength of an algorithm and the consequent retrieval effectiveness arising from its use".

Altering the emphasis slightly, significance tests showed that none of the conflation algorithms tested was significantly worse, and several were significantly better, than use of unstemmed words.

3.5 Stemming in online catalogues

As mentioned in 2.4.1, we do not know of any catalogue accessing a general collection which uses automatic stemming.

Among specialised or experimental catalogues, there is CITE, which uses a stemming procedure designed for medical terminology [19]. For the intermediate version of Okapi we used a slightly modified version of Porter's algorithm [2]. This system was not put out for live use, but experiments involving the repetition of real searches from transaction logs showed that even this allegedly 'understemming' procedure could cause serious loss of precision. The often-cited example of "communism" and "communication" becoming conflated is a good enough reason for rejecting the unconditional use of even a comparatively weak procedure.

The dangers of uninhibited stemming in online catalogues seem to arise from two causes - the general coverage of the typical academic or public library database, and the lack of specificity of many of the searches. Further, many users do not want an exhaustive search. They want to find one or two relevant items, and are not prepared to look at dozens of irrelevant records before they find them.

Of the sample searches in Table 2.1, strong stemming would adversely affect at least two: *radio* (*radiology* etc) and *modernism* (*modern*). On the other hand, some degree of stemming would benefit at least six of these searches (including *modernism/modernist*).

3.5.1 *Choice of stemming procedure for online catalogues*

Choice will be determined by the need to apply varying degrees of stemming, and by libraries' generally limited computational resources.

Of the methods described in this chapter, MARS (3.3.8) looks the most ambitious. It also looks difficult to implement and computationally demanding.

An iterative procedure is attractive because it should need a smaller dictionary (long suffixes are treated as a sequence of shorter ones to be successively removed). It would usually be possible to partition both iterative and longest match procedures into two or more stages.

(It is worth noting that the degree of compression which stemming produces is irrelevant. Even if stemming reduces the number of entries in an index by half the total storage requirement is almost unaffected. Nearly all of the word index to a large file is made up of postings, or pointers to the records which are indexed by the words.)

We chose Porter's algorithm for the intermediate Okapi because it is short, simple, easy to program and readily available. It makes about four kilobytes of code and data in Z80 assembly language. We saw no reason to alter our choice for this project, and merely split the procedure into two stages, the first performing "weak" stemming and the two combined performing "strong stemming". We incorporated some "spelling standardisation" into the "weak" stage. This is described in Chapter 6.

References

1 MATTHEWS J R. *Public Access to Online Catalogs : a planning guide for managers*. 2nd ed. Online Inc, 1986.

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

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

4 OVERHAGE C F J and REINTJES J F. Project Intrex : a general review. *Information Storage and Retrieval 10* (5/6), May/June 1974, 157-188.

5 LOVINS J B. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* 11, 1968, 22-31.

6 LOWE T C, ROBERTS D C and KURTZ P. *Additional Text Processing for On-line Retrieval (The RADCOL System)*. (Technical report RADC-TR-73-337). 1973.

7 TARRY B D. *Automatic Suffix Generation and Word Segmentation for Information Retrieval*. M.Sc. thesis, University of Sheffield, 1978.

8 FIELD B J. *Semi-automatic Development of Thesauri using Free-language Vocabulary Analysis (Part 1 only)*. (Report no. R75/24). Inspec, 1975.

9 SALTON G. *Automatic Information Organization and Retrieval*. McGraw-Hill, 1968.

10 DATTOLA R T. FIRST : Flexible Information Retrieval System for Text. *Journal of the American Society for Information Science 30* (1), January 1979, 9-14.

11 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*. Edited by C J van Rijsbergen. Cambridge University Press *on behalf of* the British Computer Society, 1985, 409-419.

12 BELL C L M and JONES K P. A minicomputer retrieval system with automatic root finding and roling facilities. *Program 10* (1), Jan 1976, 14-27.

13 CERCONE N. A heuristic morphological analyser for natural language understanding programs. *The IEE Computer Society's First International Computer Software and Applications Conference, Chicago, Illinois, 8-11 November 1977.* New York : IEEE, 1977, 676-682.

14 CERCONE N. Morphological analysis and lexicon design for natural-language processing. *Computers and the Humanities 11*, 1978, 235-258.

15 NIEDERMAIR G Th, THURMAIR G and BUTTEL I. MARS : a retrieval tool on the basis of morphological analysis. 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.

16 DAWSON J L. Suffix removal and word conflation. *ALLC Bulletin*, Michaelmas 1974, 33-46.

17 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.

18 LANDAUER C and MAH C. Message extraction through estimation of relevance. *Research and Development in Information Retrieval. Proceedings of the ACM-BCS Symposium, Cambridge, 23-26 June 1980*. Cambridge University Press, 1980.

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