A STEMMING ALGORITHM FOR LATIN TEXT DATABASES

ROBYN SCHINKE*, MARK GREENGRASS*, ALEXANDER M. ROBERTSON† *and* PETER WILLETT†‖

Humanities Research Institute and Departments of History and of Information Studies†, University of Sheffield, Western Bank, Sheffield S10 2TN*

This paper describes the design of a stemming algorithm for searching databases of Latin text. The algorithm uses a simple longest-match approach with some recoding but differs from most stemmers in its use of two separate suffix dictionaries (one for nouns and adjectives and one for verbs) for processing query and database words. These dictionaries and the associated stemming rules are arranged in such a way that the stemmer does not need to know the grammatical category of the word that is being stemmed. It is very easy to overstem in Latin: the stemmer developed here tends, rather, towards understemming, leaving sufficient grammatical information attached to the stems resulting from its use to enable users to pursue very specific searches for single grammatical forms of individual words.

INTRODUCTION

AN IMPORTANT COMPONENT of any system for text searching is the ability to identify accurately the variant word forms that arise from grammatical modifications or alternative spellings of the words in a user's query. Such variants are normally encompassed by means of either right-hand truncation or stemming. Truncation is carried out by the searcher, who removes as many letters from the right-hand side of the word as seems appropriate to achieve a plausible root, and the search then retrieves all words that commence with this root, regardless of their endings. Stemming, conversely, is carried out automatically by reducing all words with the same stem to a common form, typically by removing the inflectional and derivational suffixes [1–3]. Stemming algorithms for English have been discussed extensively in the literature [1–6] and there has recently been much interest in stemming algorithms for other languages [7–12]. In this paper, we describe the development of a stemming algorithm to support free-text searches of Latin databases, an increasing number of which have become available to scholars over the last few years.

In some respects, Latin is ideally suited to right-hand truncation, since it is an inflected language which makes extensive use of suffixes to convey syntactic, rather than semantic, information about words. In practice, however, a simple, right-hand truncation search for a standard Latin word produces very poor

‖To whom all correspondence should be addressed. Email address: *p.willett@sheffield.ac.uk*

Journal of Documentation, vol. 52, no. 2, June 1996, pp. 172–187

results for two principal reasons. The first problem is that many Latin words have more than one distinct stem. For example, nouns such as *vox* (voice) yield the stems 'vox-' and 'voc-', while verbs such as *agere* (to do, drive) exhibit a minimum of three stems: 'ag-', 'eg-', and 'act-'. Accordingly, high recall will only be achieved in a right-hand truncation search if the user knows all of the stems of the query word and carries out several separate, but related, searches of the database for each distinct query word. A further problem is that a simple truncation search produces a large number of words that are unrelated to the initial query. This arises not only because Latin roots tend to be quite short (e.g. *vox* and *agere* as discussed above) but also because many Latin words with different meanings have similar roots, e.g. the three words *portus* (harbour), *porta* (gateway) and *portare* (to carry).

This combination of factors means that when Latin words are truncated to their linguistic roots, such as *ducere* to 'duc-' or *mater* to 'matr-', the resultant stems are so short that many other words may begin with the same three or four letters, and many of these words may not be semantically related to the query word. For example, when the verb *ducere* (to lead) was sought in the *Patrologia Latina* (a database that contains a huge compendium of patristic and medieval thought originally published in the middle of the nineteenth century) with the stem 'duc-', a total of 404 different words was retrieved. Only 118, just 29%, at most, of these words can actually be variants of the query verb since Latin verbs have a maximum of 144 different forms and since twenty-six of the forms of *ducere* commence with 'dux-', rather than 'duc-'. By similar reasoning, only 7% at most of the 2,104 words retrieved in a search of the same database for the verb *fero*, *ferre*, *tuli*, *latum* (to carry), using the three stems 'fer-', 'tul-' and 'lat-', can possibly be variants of the query verb. The results are somewhat better when words with longer roots are searched. For example, searching the stem 'ambul-' for the verb *ambulare* (to walk) retrieved 129 different words, of which 94 (73%) were found to be query variants. These occurrences are examples of the general problem of overstemming, which occurs when too short a stem remains after the removal of a word's ending(s), with the result that unrelated words are conflated to the same stem. The converse of this, understemming, occurs when too short a suffix is removed (so that related words are not all conflated to the same stem). Both problems occur with any stemming algorithm [13, 14]; in the particular context of Latin, we believe that understemming is of less importance since it can be circumvented, at least in part, by an appropriate algorithm design, as detailed below.

Right-hand truncation *can* be used to search Latin text databases, but only if the user has sufficient knowledge of Latin to be able to enter all of the necessary stems for a given query word and to identify the (possibly large number of) related forms in the search output. In this paper, we describe a stemming algorithm for Latin that seeks to provide effective access to such databases for users with only a basic knowledge of the language. The next section provides a brief overview of the main characteristics of stemming algorithms, and we then discuss some of the main features of Latin that need to be reflected if a stemmer is to perform effectively. The following, and longest, section details the

development of our stemmer, building on an initial extremely simple procedure in several stages to give a final algorithm that does, we believe, provide an effective means of processing Latin dictionaries. The final section contains our initial thoughts on how the stemmer will be implemented within a retrieval system to allow the searching of Latin texts; this implementation will be discussed in a future paper.

STEMMING ALGORITHMS

The rationale for stemming is that, for many languages, the semantically-useful information in a word is in the stem and the suffixes merely change its grammatical form. If this is correct, then words which are structurally similar will also be related in meaning, and stemming may thus be expected to increase the effectiveness of an information retrieval system. Stemming is also used in natural language processing, where the identification of a particular suffix may assist parsing, and textual studies of, for example, the grammatical structure of a particular author's writings.

A stemmer typically contains a list of rules, each consisting of a suffix and the conditions under which it may be removed from a word. There are two principal methods of applying the suffix rules. The more common is a longest-match algorithm, which involves the removal of the longest suffix in the dictionary that matches the ending of the query word. Alternatively, in an iterative algorithm, suffixes are removed one by one in sequence. Thus, if a suffix in the dictionary is found in the query word the former is removed and the word is then checked again for the presence of further suffixes, with any more that are found being removed also. Iteration is based on the observation that suffixes are generally attached to the root in a certain order and they can hence be removed in reverse order, e.g. the series of words *computer*, *computerise* and *computerisation*. A stemmer may allow recoding to deal with spelling exceptions, i.e. instances of roots varying in spelling depending upon the suffixes that originally followed them (e.g. the words *absorb* and *absorption*). Rules may also be included in a stemmer to prevent words being shortened excessively (e.g. to prevent the removal of '-ing' from *ring* or '-as' from *gas*), with the possibility of specifying different minimum stem lengths for different roots.

The English stemmer that is most widely used in information retrieval is probably that described by Porter [4], who argues that there is a stage in a suffix-stripping algorithm where the addition of more rules giving an improvement in one area will cause an equal degradation elsewhere. A large number of rules can make a stemmer unnecessarily complex and his algorithm hence uses only a limited number of suffixes, organised into five main groups that are checked in sequence by an iterative procedure. A novel feature of the algorithm is that it characterises a word by its measure, which defines the occurrence of consonant-vowel substrings within a word and which constrains the suffixes that are available for removal at any stage of the processing of an input word. The algorithm is simple in concept and highly efficient, and has been found to work well in practice [2, 4]. Porter's algorithm, like many others, has been designed

for processing general-purpose English text; an example of a special-purpose stemmer is that described by Ulmschneider and Doszkocs [5] for processing medical texts.

Stemming techniques can be applied to any language where the semantic information is contained in the stem and where variant word forms are created by adding suffixes to this stem, and the last few years have seen several reports of the development of stemmers for languages other than English. Popovic and Willett [7] describe the implementation of a stemmer for Slovene that contains no less than 5,276 suffixes, each with its minimum stem length and one of eight action codes which determine which of a list of context-sensitive rules should be applied, and three different sets of recoding rules. French is another language with a more complex morphology than English. This complexity is handled in the stemmer described by Savoy [8] by means of grammatical categories. The morphological analysis involves the use of a declension file, with the declension for a particular word being identified from its terminal letter. In addition, there are rules for derivational suffixes and some recoding rules. Kalamboukis [10] presents a stemmer for modern Greek, a language with forty-one different declensional categories, that covers both inflectional and derivational morphology, but excludes the problem of accents. This algorithm is of interest since, like the algorithm described in this paper, inflectional suffixes are processed separately for different parts of speech (such as nouns, adjectives and verbs). Still more complex procedures are required for stemming agglutinative languages such as Turkish, where very large numbers of suffixes can be added to a basic root [11], or for stemming languages such as Malay, which involve extensive prefixing as well as suffixing [12].

There have been several studies that have evaluated the extent to which the use of a stemmer can increase the effectiveness of retrieval from large text databases. The evidence to date suggests that stemming will not bring about significant increases in performance for English [15, 16] but that such improvements can occur with other, more complex languages [10, 17].

INITIAL DESIGN CONSIDERATIONS

Latin words may readily be grouped into broad categories according to their part of speech and general form, and it is thus simple to define a list of suffixes that may be removed from any particular word. Nouns, for example, are commonly grouped into five declensions, each with reasonably distinct endings (as exemplified by the second declension endings shown in Figure 1) and most adjectives use the suffixes of these five declensions. It is thus easy to compile a list of twenty-seven distinct suffixes that should be removed from nouns and adjectives. However, additional particles are added to Latin adjectives when they are used in comparisons. For example, the adjective *benignus*, *benigna*, *benignum* (kind) becomes *benignior*, *benignius* (kinder), and *benignissimus*, *benignissima*, *benignissimum* (kindest). As a result, twenty-two additional suffixes must be included if all adjectival forms are to be reduced to their appropriate stems. Finally, certain types of verb forms, such as present and future participles,

Case	*Singular ending*	*Plural ending*
Nominative	-us (-um)	-i (-a)
Vocative	-e (-um)	-i (-a)
Accusative	-um	-os (-a)
Genitive	-i	-orum
Dative	-o	-is
Ablative	-o	-is

FIGURE 1. *Second declension noun endings*

gerunds, and forms of the gerundive, are actually treated as if they were nouns and adjectives: these require the use of a further thirty-five suffixes if these forms are to be reduced to their appropriate verbal stems. These additions result in a final list of eighty-four suffixes for stemming nouns, adjectives and verbal nouns and adjectives.

The case of Latin verbs is very similar to that of nouns and adjectives, in that the basic list of suffixes which should be removed from verb forms is relatively small. All verb forms terminate in one of only thirteen different suffixes, regardless of their tense or mood, but additional particles are added to indicate various verb tenses, thus increasing the number of suffixes that may need to be removed. This behaviour is exemplified in Figure 2, which lists all of the possible forms for first-conjugation verbs such as *amare* (to love). When all of these additional particles are taken into consideration, a minimum of ninety-four suffixes must be removed from verbs to ensure that forms of all tenses are stemmed, and no less than 262 suffixes would have to be removed in order to reduce verbs of all tenses and moods to their correct linguistic roots. When these totals are combined with the suffixes which must be removed from nouns and adjectives, one obtains a list of either 178 or 346 suffixes that should be removed from Latin words in order to stem them correctly.

Although it is straightforward to compile such a list, its use is likely to result in widespread overstemming for the reasons that we have discussed previously. Indeed, it is possible to predict the types of words which would be likely to be overstemmed using such a list. For example, if the suffixes '-bam', '-bas', '-bat', '-bamus', '-batis', and '-bant' are removed from verb forms in the imperfect tense, then overstemming may occur in present-tense verb forms with roots that terminate in '-b-', e.g. the present-tense verb form *bibamus* (let us drink) would be stemmed to 'bi-' (instead of the correct form 'bib-'). The overstemming of certain present-tense forms would also result from the removal of suffixes related to the future tense such as '-bit' and '-bimus', e.g. the word *scribit* (he writes) would be stemmed to 'scri-' instead of 'scrib-'. Similarly, certain third-declension noun forms with stems ending in '-or-' or '-ar-' will not be stemmed correctly if the suffixes '-orum' and '-arum' are removed, e.g., the words *nectarum* (of the nectars) and *laborum* (of the labours) would be overstemmed to 'nect-' and 'lab-', instead of the correct stems 'nectar-' and 'labor-'.

Tense	*Voice*	*Mood*	*Stem*	*Medial vowel*	*Extra particle*	*Possible suffixes*
Present	Active	Indicative	am-	-a-		o, s, t, mus, tis, nt
Present	Passive	Indicative	am-	-o- -a-		r, ris, tur, mur, mini, ntur
Present	Active	Subjunctive	am-	-e-		m, s, t, mus, tis, nt
Present	Passive	Subjunctive	am-	-e-		r, ris, tur, mur, mini, ntur
Imperfect	Active	Indicative	am-	-a-	-ba-	m, s, t, mus, tis, nt
Imperfect	Passive	Indicative	am-	-a-	-ba-	r, ris, tur, mur, mini, ntur
Imperfect	Active	Subjunctive	amare-			m, s, t, mus, tis, nt
Imperfect	Passive	Subjunctive	amare-			r, ris, tur, mur, mini, ntur
Future	Active	Indicative	am-	-a-	-bi/bo-	s, t, mus, tis, nt
Future	Passive	Indicative	am-	-a-	-bi/bo-	r, ris, tur, mur, mini, ntur
Perfect	Active	Indicative	amav-	-i-		sti, t, mus, stis, nt
Perfect	Passive	Indicative	amat-			noun endings plus form of to be
Perfect	Active	Subjunctive	amav-		-eri-	m, s, t, mus, tis, nt
Perfect	Passive	Subjunctive	amat-			noun endings plus form of to be
Pluperfect	Active	Indicative	amav-		-era-	m, s, t, mus, tis, nt
Pluperfect	Passive	Indicative	amat-			noun endings plus form of to be
Pluperfect	Active	Subjunctive	amav-		-isse-	m, s, t, mus, tis, nt
Pluperfect	Passive	Subjunctive	amat-			noun endings plus form of to be
Future Perfect	Active	Indicative	amav-		-eri-	m, s, t, mus, tis, nt
Future Perfect	Passive	Indicative	amat-			noun endings plus form of to be

FIGURE 2. *Verb structures for the first-conjugation verb* amare *(to love)*

In practice, Latin verbs can generally be reduced to three principal components: the stem, a medial vowel, and a suffix (although an extra particle may also be required in some cases, as shown in Figure 2). It is hence possible to avoid overstemming many verb forms by removing only the suffixes which indicate the verb's person and voice. Of course, removing only these suffixes results in many verb forms being understemmed, rather than overstemmed, e.g. all of the forms of the verb *ducere* would be reduced to a total of nine distinct stems, rather than to one. We believe that understemming is almost inevitable if we are to avoid the conflation of similar stems from different Latin words that has been discussed previously. In addition, the longer stems include grammatical information about the words which may eventually be used to initiate more specific types of queries, such as searches for particular tenses of verbs, or for certain forms of adjectives. For example, by retaining at least some grammatical information after the removal of a suffix, it is possible to distinguish between the noun *portus* (harbour) and the verb *portare* (to carry). These words share the same linguistic root of 'port-'; however, an algorithm that removes a minimal number of suffixes from Latin words will stem *portus* to 'port-', and *portare* to 'porta-', 'porte-', 'portav-' 'portaba-', 'portabi-', and several other stems, all of which are longer than 'port-'.

The principal reason for using stemmers in information retrieval is to increase the recall of a search. It has thus been the case that most, though not all [2, 14], of them have tended to err on the side of overstemming, rather than understemming, so as to maximise the amount of relevant material that is retrieved in response to broadly defined, recall-oriented searches of a database. While such general searches may be carried out on Latin databases, the nature of the stored text implies that most users will be academic scholars – classicists, historians or philologists – who wish to retrieve texts that can answer highly specific questions, e.g. the extent to which a particular poet uses a particular phrase, the ways in which the meaning of a particular word differs in texts from different periods, or the precise usage of a particular word or concept. In such cases, we think that understemming is strongly to be preferred to overstemming.

DEVELOPMENT OF THE STEMMER

Initial experiments

Our initial algorithm contained a total of fifty-six suffixes: twenty-five of these related to Latin nouns and adjectives and thirty-one to verbs, as detailed in Figure 3. The initial test sample (sample A) consisted of 1,418 Latin words, comprising a manufactured list of all of the possible forms of various types of Latin nouns, adjectives and verbs. Care was taken to ensure that representative words from all of the major declensions and conjugations were present in the test sample, as well as words known to be irregular and potentially difficult for an algorithm to stem correctly, e.g. *esse* (to be), *ire* (to go), *duo* (two), *renes* (kidneys) and *res* (thing).

In the first full test of the algorithm, only 66% of the words in sample A were stemmed correctly. The great majority of the errors were due to overstemming,

-arum	-ebus	-ibus	-orum	-ius	-uum	-ae	-am
-as	-ei	-em	-es	-ia	-is	-os	-ua
-ud	-ui	-um	-us	-a	-e	-i	-o
-u							

(a)

-iuntur	-beris	-erunt	-untur	-mini	-ntor	-ntur	-stis
-tote	-bor	-ero	-mur	-mus	-nto	-ris	-sti
-tis	-tor	-tur	-iunt	-unt	-bo	-ns	-nt
-ri	-te	-to	-m	-r	-s	-t	

(b)

FIGURE 3. *Initial list of suffixes for (a) nouns/adjectives and (b) verbs*

which was principally caused by the removal of suffixes relevant to verb forms from nouns and adjectives (and, to a lesser extent, the removal of suffixes relevant to noun forms from verbs). Many of these verb suffixes began with consonants, such as '-tis' and '-mus', and often corresponded to suffixes relevant to nouns and adjectives, such as '-is' and '-us' plus the final consonant of a noun or adjective stem. For example, when the suffix '-tis' was removed from the verb form, *amatis* (you love) was stemmed correctly to 'ama-', but the noun *dignitatis* (of honour) was overstemmed to 'dignita-' (instead of the correct stem 'dignitat-').

Use of two sets of rules

The need to avoid removing verb endings from nouns and adjectives and to avoid removing noun/adjective endings from verbs led us to include two separate sets of suffixes (one for verb forms and one for noun/adjective forms) in the algorithm, resulting in the creation of two separate dictionaries of stemmed words when the algorithm is applied to a file of text. The first set of rules removes the suffixes associated with the five declensions of nouns and adjectives, and the second set of rules removes the suffixes associated with the four conjugations of verbs, including deponent verbs. The algorithm is applied to an input query word using first the set of noun suffixes and then the set of verb suffixes. The result is that one dictionary contains a list of words in which nouns and adjectives are stemmed correctly, while any other words are either not stemmed at all or have been stemmed in such a way that they could not be confused with nouns and adjectives. For the second dictionary, the converse applies: verb forms are stemmed correctly, while nouns and adjectives are processed so that they cannot possibly be confused with the verb stems. The aim of this approach is to ensure that the two main classes of words are stemmed in an appropriate manner but without the need for the extensive linguistic processing that would be required to identify the parts of speech for each of the words in a text that was to be stemmed.

When the new algorithm was tested, the results were far better than when all of the suffixes were removed from all of the words. Indeed, when sample A was used, no less than 99% of the nouns and adjectives, and 93% of the verb forms,

were stemmed correctly. However, when the algorithm was applied to a second sample of text, sample B, which consisted of 439 distinct words from a few, short Latin poems in the Hartlib Papers [18, 19], the figures were effectively reversed, with 92% of the nouns and 99% of the verbs being stemmed correctly.

The reason for the drop in the algorithm's effectiveness in stemming nouns, and its concomitant improvement in stemming verbs, may be explained by two unrelated phenomena. However, the root cause of the differences in performance undoubtedly lies in the fact that sample B consists of entire Latin sentences, written to convey thoughts and feelings, while sample A is merely a list of carefully selected individual Latin words. For instance, sample A contains several verbs with irregular forms and short stems, such as *esse* (to be), *ire* (to go) and *agere* (to do), which the algorithm was unable to stem very accurately. On the other hand, in the poems of sample B, the verbs used were more descriptive words appropriate to the genre, with longer stems and fewer irregular forms. Examples of such words include *occumbere* (to fall in death), *temnere* (to despise) and *extimescere* (to dread), all of which the algorithm was able to stem correctly.

Neither the length of the stems nor the irregularity of the words could have contributed to the decrease in the number of nouns which were stemmed correctly in sample B, since most nouns have stems which are at least three characters long, and since there are very few nouns which deviate from the patterns set by the five declensions. In this case, the problem was caused by the fact that when Latin is written in full sentences as literature, like the poems from the Hartlib Papers, authors tend to add certain suffixes to words in addition to their usual endings, specifically the practice of adding enclitic suffixes to the ends of words instead of using conjunctions. The identification of this problem led to the next modification of the basic algorithm.

Processing of enclitic suffixes

There are just three enclitic suffixes: '-que', which is used instead of 'and'; '-ne', which is used to indicate a question; and '-ve', which is used in place of 'or'. For example, instead of writing *pueri et puellae* (the boys and the girls), authors may write *pueri puellaeque* to give the same meaning. In such a case, the addition of the terminal '-que' effectively hides the suffix '-ae', which ought to be removed from the stem 'puell-'.

It is possible to remove the three enclitic suffixes from all words before stemming starts but such a simplistic approach would lead to widespread overstemming, since many words incorporate these syllables as a part of their stems, e.g. *agmine* (from *agmen*: battle line), *nive* (from *nix*: snow) and *torque* (from *torquere*: to twist), to name but a few. An analysis of a larger group of Latin documents from the Hartlib Papers (the sample C that is described below) showed that only 6% of words ending in '-ve' and 4% of words ending in '-ne' actually included an enclitic suffix, which would have to be removed in order for the word to be stemmed correctly. It was hence decided to ignore the problems caused by these two enclitic suffixes and not to attempt to remove them from words prior to stemming. However, the same analysis showed that

atque, quoque, neque, itaque, absque, apsque, abusque, adaeque, adusque, denique, deque, susque, oblique, peraeque, plenisque, quandoque, quisque, quaeque, cuiusque, cuique, quemque, quamque, quaque, quique, quorumque, quarumque, quibusque, quosque, quasque, quotusquisque, quousque, ubique, undique, usque, uterque, utique, utroque, utribique, torque, coque, concoque, contorque, detorque, decoque, excoque, extorque, obtorque, optorque, retorque, recoque, attorque, incoque, intorque, praetorque

FIGURE 4. *List of words from which the enclitic suffix '-que' should not be removed in Step 3 of the algorithm shown in Figure 7*

it was necessary to remove the enclitic suffix from no less than 83% of words ending in '-que' before those words could be correctly stemmed. It is thus clear that this enclitic suffix is very widely used and that it cannot simply be removed whenever it occurs at the end of a word (since this would still leave almost a fifth of the words incorrectly stemmed). We have thus created a list of words from which '-que' should *not* be removed prior to stemming. This list, which is shown in Figure 4, includes the most common conjunctions and query words (such as *atque* (and), *quoque* (also) and *quandoque* (at what time soever) as well as verbs of the second conjugation with stems in 'qu-' that will end in '-que' in the singular imperative (such as *torque* (twist) and *coque* (cook)).

Once these changes had been made to the algorithm, the percentage of nouns in sample B which were stemmed correctly rose from 92% to 98%, while the percentage of verbs stemmed correctly remained the same. These results demonstrate clearly the need to take account of enclitic suffixes when stemming Latin.

The resulting algorithm was then applied to a much larger file drawn from the Hartlib Papers collection. This test sample (sample C) contained forty-nine complete documents, which represent 5% of the total Latin documents in the collection and which contain a total of 16,180 distinct words. The selected documents cover a wide range of topics (including philosophy, astrology, medicine, religion and linguistics, as well as a variety of personal letters) written by at least twenty-six different authors, and can thus be regarded as typical of neo-classical Latin. Application of the algorithm to sample C gave an overall success rate of 99%, suggesting that any further enhancements of the algorithm should be restricted to small, well defined groups of words and word endings since there was clearly little scope for substantial further increases in performance.

Minor enhancements

Thus far, the minimum stem length, i.e. the minimum number of characters which must remain after the removal of a suffix, had been set at three characters. Changing this to two characters did not greatly improve the overall results but did permit the correct stemming of a limited number of very short Latin words, and the procedure was hence deemed to be worthwhile. Examples that were now successfully stemmed included *dies* (day), *via* (road) and *meus* (my).

The next modification tackled a small but important characteristic of neo-classical Latin texts: the common use of the letter 'j' in place of the letter 'i'. Although the Romans never used the letter 'j' at all, centuries of linguistic development led to the view that the use of the letter 'i' before another vowel should be considered as a 'consonantal i' that was generally replaced by the letter 'j'. For example, where the Romans wrote words such as *Iulius*, *coniunx* (wife) and *iam* (now), later scholars often wrote *Julius*, *conjunx* and *jam*. The algorithm was hence modified to transform all occurrences of the letter 'j' to the letter 'i' prior to stemming. A similar difficulty was found to exist with the letters 'v' and 'u', which are often used interchangeably in neo-classical Latin, and an analogous replacement strategy was hence adopted. It should be noted that, although the introduction of the distinct letters 'j' and 'u' did not occur until after the period during which most classical Latin literature was written, many such texts make use of both of these letters in modern editions. The use of only one letter from each of these pairs thus enables the algorithm to stem Latin texts of all periods: classical, medieval, Renaissance and modern.

Although the algorithm was able to stem Latin verbs with 99% accuracy at this stage, the program still reduced the 144 verb forms to at least nine separate stem classes. In order to determine whether this number of classes might be reduced, the algorithm was altered to allow for the secondary removal of the intermediate particles which indicate verb tenses (as shown for the first conjugation of verbs in Figure 2). When the program had already removed one of the thirteen suffixes used to indicate a verb form's person and number, it subsequently removed three particles, and transformed two others. The particles '-ba-', '-bi-', and '-sse-', which indicate the imperfect, future, and pluperfect (subjunctive) tenses, respectively, were removed completely. The particles '-era-' and '-eri-', which indicate the pluperfect (indicative) and perfect (subjunctive) tenses, respectively, were both transformed to the letter 'i'. These changes did in fact reduce the number of distinct verb stems to five, but caused some degree of overstemming of words with stems including these particles, e.g. the word *conferam* (I will consider) was overstemmed to 'confi-' instead of correctly to 'confera-'. It was hence decided not to include secondary particle removal in the algorithm.

The final algorithm

The rules used for stemming nouns and adjectives remain separate from those rules used to stem verb forms. In both cases, suffixes are removed using a longest-match procedure in which just the longest suffix that matches the end of an input word is removed. No stemming takes place if a match cannot be obtained with any of the suffixes in either of the suffix dictionaries. By structuring the algorithm in this way, it is possible to ensure that entire classes of words in the stemmed dictionaries do not need to be processed by the retrieval routines (which will be described more fully in a subsequent communication). For example, since the set of rules which removes suffixes that are common to nouns and adjectives does not include any suffixes which are relevant to verbs, verb forms such as

portat (he carries), *ducimus* (we lead), and *legunt* (they choose) remain intact in the dictionary which these rules create. These words thus cannot be reduced to any stem which might be identical to the stem of a noun or adjective. Likewise, the set of rules which removes suffixes relevant to verb forms either allows nouns and adjectives to remain intact, or stems them in such a way that words such as *portis* (to the gates), *ducibus* (by the leaders) and *legum* (of laws) cannot produce stems which are identical to those of verb forms. This situation is exemplified by Figure 5, which contains a range of word types and the stems that resulted from application of the final version of the algorithm.

Unstemmed	*Noun-based stem*	*Verb-based stem*
Apparebunt	Apparebunt	Apparebi
Aquila	Aquil	Aquila
Colluxisset	Colluxisset	Colluxi
Deprehendebatur	Deprehendebatur	Deprehende
Dexisse	Dexiss	Dexi
Ducibus	Duc	Ducibu
Ducimus	Ducim	Duci
Elucidatione	Elucidation	Elucidatione
Fratre	Fratr	Fratre
Fratrem	Fratr	Fratre
Fratres	Fratr	Fratre
Fratri	Fratr	Fratr
Fratrum	Fratr	Fratru
Legum	Leg	Legu
Legunt	Legunt	Legi
Libertas	Libert	Liberta
Libertate	Libertat	Liberta
Libertatem	Libertat	Libertate
Libertates	Libertat	Libertate
Libertatis	Libertat	Liberta
Mathematica	Mathematic	Mathematica
Mathematici	Mathematic	Mathematici
Mathematicum	Mathematic	Mathematicu
Nobilissima	Nobilissim	Nobilissima
Nobilissimam	Nobilissim	Nobilissima
Nobilissime	Nobilissim	Nobilissime
Nobilissimo	Nobilissim	Nobilissimo
Nobilissimum	Nobilissim	Nobilissimu
Portat	Portat	Porta
Portis	Port	Por

FIGURE 5. *Examples of words and the corresponding stems produced by the final version of the stemming algorithm*

The final set of rules for stemming nouns and adjectives contains nineteen suffixes while the set for stemming verbs contains twenty-five suffixes, as shown in Figures 6(a) and 6(b), respectively. Nine of the twenty-five verb suffixes are transformed into other endings, rather than being removed directly. For example, the suffixes '-iuntur', '-erunt', '-untur', '-iunt', and '-unt' are all changed to the shorter suffix '-i', a transformation that effectively normalises certain irregular verb forms in the present and perfect tenses. Further, the suffixes '-beris', '-bor', and '-bo', are changed to '-bi' (allowing the conflation of all verb forms of the future tense) and the suffix '-ero', a slightly irregular suffix of the future perfect tense, is changed to '-eri'. These modifications were carried out to ensure that all verb forms of the same tense are reduced to a common stem, thus allowing for more specific types of query to be searched.

The overall structure of the final algorithm is shown in Figure 7. It has been implemented in the C programming language, using some of the data structures and character manipulation routines in the implementation of Porter's English stemmer that is given as an appendix to the paper by Frakes [3]. The performance of the algorithm is exemplified by the sets of words and stems shown in Figure 5. While there are still some limitations in the stemmer that require further attention, e.g. it has not been applied to vulgar Latin and no account is taken of the effects of accentuation on words, we believe that it already provides a powerful means of identifying a large fraction of the morphological variants of Latin words.

CONCLUSIONS

Truncation and stemming are the most common approaches to term conflation in free-text retrieval systems. Conventional right-hand truncation is inappropriate for searching Latin texts since most Latin words have more than one distinct stem and since many Latin words with different meanings may have very similar roots. The stemming algorithm described in this paper seems to provide an effective way of surmounting these two problems, as demonstrated by the results shown in Figure 5.

-ibus	-ius	-ae	-am	-as	-em	-es	-ia
-is	-nt	-os	-ud	-um	-us	-a	-e
-i	-o	-u					

(a)

-iuntur	*-beris*	*-erunt*	*-untur*	*-iunt*	-mini	-ntur	-stis
-bor	*-ero*	-mur	-mus	-ris	-sti	-tis	-tur
-unt	*-bo*	-ns	-nt	-ri	-m	-r	-s
-t							

(b)

FIGURE 6. *Final list of suffixes for (a) nouns/adjectives and (b) verbs (where suffixes that are transformed, rather than removed, appear in italic type)*

1. Read the next word to be stemmed.
2. Convert all occurrences of the letters 'j' or 'v' to 'i' or 'u', respectively.
3. If the word ends in '-que' then
   if the word is on the list shown in Figure 4 then
   write the original word to both the noun-based and verb-based stem dictionaries and return to Step 1
   else remove '-que'.
4. Match the end of the word against the suffix list shown in Figure 6(a), removing the longest matching suffix (if any).
5. If the resulting stem contains at least two characters then write this stem to the noun-based stem dictionary
   else write the original word to the noun-based stem dictionary.
6. Match the end of the word against the suffix list shown in Figure 6(b), identifying the longest matching suffix (if any).
   If any of the following suffixes are found then convert them as shown:
   '-iuntur', '-erunt', '-untur', '-iunt', and '-unt' to '-i';
   '-beris', '-bor', and '-bo' to '-bi';
   '-ero' to '-eri'
   else remove the suffix in the normal way.
7. If the resulting stem contains at least two characters then write this stem to the verb-based stem dictionary
   else write the original word to the verb-based stem dictionary.
8. Return to Step 1 if there are more words to be processed.

FIGURE 7. *The final version of the stemming algorithm*

There are two major features of our algorithm that differentiate it from other stemmers that have been described in the literature [1–12]. Firstly, the algorithm generates two stem dictionaries. This it does by using two sets of stemming rules which keep nouns and adjectives separate from verb forms by default, but without the need to encode the parts of speech for the words that are to be stemmed. Secondly, the policy of deliberately understemming many words leaves enough grammatical information attached to the resultant word stems to enable the algorithm to distinguish easily between different words with similar roots. In addition, this feature will enable users to pursue very specific searches for single grammatical forms of some types of words, an important requirement for the intended users of the procedure.

Thus far, we have considered only the use of the stemmer for processing word dictionaries, without saying anything about how the resulting stem dictionaries will be searched. Because we hope that even scholars with limited expertise in Latin (and probably even less expertise in term conflation techniques) will wish to use the search software, we have chosen to require that they will enter only the forms of words that are prominently listed in any standard Latin dictionary. These standard forms provide a natural search mechanism since they are one of the principal means by which students first learn the language. Our initial experiments have demonstrated the general validity of this dictionary-based search procedure, the details of which will be presented in a subsequent paper.

When the query word is a noun or an adjective, the user will enter two forms: the nominative and genitive singular. For example, the word *puella* (girl) will be entered as 'puella, puellae', while the word *dux* (leader) will be entered as 'dux, ducis'. When the query word is a verb, the user will enter four forms: the present form, the present infinitive, the perfect form, and the past participle. For example the verb *amare* (to love) will be entered as 'amo, amare, amavi, amatum', and the verb *ferre* (to carry) will be entered as 'fero, ferre, tuli, latum'. In this way, the needs of the user who is unfamiliar with Latin grammar will be served, since the query will involve just those word forms that are explicitly listed in a standard Latin dictionary. Further, this query entry procedure will surmount one of the most significant problems facing the accurate retrieval of words from Latin texts, *viz* the fact that most Latin words have multiple stems. Since Latin dictionaries always present words with the minimum number of their forms needed to generate all of their possible grammatical variants, requiring the users to type in the exact dictionary entry is not only much simpler for them but also provides the stemmer with all of the information needed to identify every variant of the query word. In fact, the search procedure can be automated in large part if a Latin dictionary is available in machine-readable form. In this case, the user can specify just a single query form, such as *amare*, use the dictionary to identify the other necessary principal parts and then generate the query for the retrieval system by means of a cut-and-paste operation.

ACKNOWLEDGEMENTS

We thank the British Library Research and Development Department and the Library of the University of Sheffield for funding.

REFERENCES

1. LOVINS, J.B. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, *11*, 1968, 22–31.
2. LENNON, M., PIERCE, D.S., TARRY, B.D. *and* WILLETT, P. An evaluation of some conflation algorithms for information retrieval. *Journal of Information Science*, *3*, 1981, 177–183.
3. FRAKES, W.B. Stemming algorithms. *In*: FRAKES, W.B. *and* BAEZA-YATES, R., *eds. Information retrieval: data structures and algorithms*. Englewood Cliffs: Prentice-Hall, 1992, 131–160.
4. PORTER, M.F. An algorithm for suffix stripping. *Program*, *14*, 1980, 130–137.
5. ULMSCHNEIDER, J.E. *and* DOSZKOCS, T. A practical stemming algorithm for online search assistance. *Online Review*, *7*, 1983, 301–308.
6. PAICE, C.D. Another stemmer. *ACM SIGIR Forum*, *24*(3), 1990, 56–61.
7. POPOVIC, M. *and* WILLETT, P. Processing of documents and queries in a Slovene language free text retrieval system. *Literary and Linguistic Computing*, *5*, 1990, 182–190.
8. SAVOY, J. Stemming of French words based on grammatical categories. *Journal of the American Society for Information Science*, *44*, 1993, 1–9.

9. AL-KHARASHI, I.A. *and* EVENS, M.W. Comparing words, stems and roots as index terms in an Arabic information retrieval system. *Journal of the American Society for Information Science*, *45*, 1994, 548–560.
10. KALAMBOUKIS, T.Z. Suffix stripping with modern Greek. *Program*, *29*, 1995, 313–321.
11. SOLAK, A. *and* OFLAZER, K. Design and implementation of a spelling checker for Turkish. *Literary and Linguistic Computing*, *8*, 1993, 113–130.
12. AHMAD, F., YUSOFF, M. *and* SEMBOK, T.M.T. Experiments with a Malay stemming algorithm. Submitted for publication.
13. LOVINS, J.B. Error evaluation for stemming algorithms as clustering algorithms. *Journal of the American Society for Information Science*, *22*, 1971, 28–40.
14. PAICE, C.D. An evaluation method for stemming algorithms. *In*: CROFT, W.B. *and* VAN RIJSBERGEN, C.J., *eds. SIGR '94 - Proceedings of the seventeenth annual international ACM-SIGR conference on research and development in information retrieval*. London: Springer-Verlag, 1994, 42–50.
15. HARMAN, D. How effective is stemming? *Journal of the American Society for Information Science*, *42*, 1991, 7–15.
16. KEEN, E.M. The effect of stemming strength on the effectiveness of output ranking. *In*: JONES, K.P., *ed. The structuring of information - Informatics 11*. London: Aslib, 1991, 37–50.
17. POPOVIC, M. *and* WILLETT, P. The effectiveness of stemming for natural-language access to Slovene textual data. *Journal of the American Society for Information Science*, *45*, 1992, 384–390.
18. LESLIE, M. The Hartlib Papers Project: text retrieval in large datasets. *Literary and Linguistic Computing*, *5*, 1990, 58–69.
19. GREENGRASS, M. Samuel Hartlib: 'intelligenceur' européen. *In: Diffusion du savoir et affrontement des idées, 1600-1770*. Festival d'Histoire de Montbrison, 1992, 213–234.

(Revised version received 13 December 1995)