Introduction: Lemmatization and parsing

By its ideal definition, lemmatization is a process wherein the inflectional and variant forms of a word are reduced to their lemma: their base form, or dictionary look-up form. When one lemmatizes a text, one replaces each individual word in that text with its lemma; a text in English which has been lemmatized, then, would contain all forms of a verb represented by its infinitive, all forms of a noun by its nominative singular, and so forth.[1]

Those involved in the production of dictionaries and concordances have long realised the value of lemmatization. When generating word indexes, concordances, and dictionaries from a text, lemmatizing that text beforehand ensures that all forms of a particular word within it can be located by searching only for its lemma form.[2] Even those involved solely as users of such a resource realize gains from the process, because every time someone looks up a headword (which is, typically, the lemma form of a word) in a standard dictionary that person is reaping the benefits of lemmatization; so, too, when using many concordances and indexes. In electronic texts, which are themselves potential dictionaries, concordances, and indexes, lemmatization ensures the accuracy of search procedures, both simple and complex alike. But the use of texts which have been lemmatized is not limited solely to such procedures. Because the numerous forms a word can take are reduced in such a text (since the lemma replaces variant forms), the overall number of individual word types in that text is decreased; thus, a lemmatized text is an invaluable aid for semantic studies and others using analysis techniques involving repeating sequences of words or word pairs, and other related studies.

The syntactic counterpart of lemmatization is parsing, an act wherein each word in a text is assigned a tag that reflects its part-of-speech and syntactic function. In a parsed text, all verb forms are marked as verbs, perhaps with varying degrees of specificity within that category (lexical, modal, primary; first person singular, first person plural, and so forth), all noun forms as nouns, and so forth; possibly, as is typical with lemmatized texts, each word is replaced by the tag which represents it, though this need not be the case.

A parsed text is more difficult to place, in terms of its everyday applications, than one which has been lemmatized in part because parsing and lemmatization have been traditionally treated as somewhat separate acts and, also, in part because the daily activities of the typical scholar do not often involve conscious contact with a text in this manner. It is true that dictionaries, as well as some concordances and indexes, do parse their texts in that they assign a syntactic marker to the words they describe, but the majority of scholars would not typically search for an occurrence of, say, a lexical verb the way one might search for the word *run*, nor are standard resources oriented to allow such searches easily to be carried out. But, just as this last statement -- "nor are standard resources oriented to allow such searches easily to be carried out" -- is less true today than it was some 10 years ago (due to the recent generation of computerised dictionaries, which allow much flexibility in searching) so is it less true today that lemmatizing and parsing should be treated as isolated acts. Of course, any project requiring the analysis of morphological and syntactic structures will clearly demand a parsed text. But work which involves, for example, stylistic analysis and authorship attribution often benefits considerably from a text which has been *both* lemmatized and parsed; so, too, can searches of large text corpuses be enhanced if that corpus has been treated by both processes. In these instances and others, texts which are both parsed and lemmatized are of considerable benefit to the scholar interested in computer-assisted textual research.[3]

1. The Role of the computer

Performing each of the two acts of lemmatizing and parsing requires a great deal of close, comparative, and

repetitive work; as those engaged in such work often find the computer to be a valuable aid, it comes as no surprise that recent research has shown that lemmatization and parsing are operations which are most efficiently carried out with the assistance of a computer. Programs such as *EYEBALL*, *MicroEYEBALL*, and the *Micro English Parser* for English, *LexiTex* and *Lemmatiseur* for French, and *LEMMA2* for German, just to name a few, exemplify advances made in his area.[4] Even with this good development, however, the two tasks are still some distance from being characterised as fully and accurately automated processes.

The complexities inherent in written language -- chiefly, the ambiguity of similar graphic forms (homographs) -- require that manual interaction, to varying but considerable degrees, is necessary when preparing any lemmatized and/or parsed text by computer-assisted means. Because levels of homography vary between languages, however, the amount of success one may hope to achieve when using software to assist in these processes is not determined solely by the software one employs: it is also determined to a large degree by the characteristics of the language with which one is working. While French, English, and German texts can fall victim to substantial errors in parsing and lemmatization because of homography, texts in languages such as Latin and Hebrew, both languages which are much more affected by this phenomenon, yield far more erroneous results from automated processes. Such errors must be corrected manually, and the possibility of error requires that all texts which have been processed with the aid of the computer must be thoroughly proofread. Thus, while many benefits can come from having and using a parsed and/or lemmatized text,[5] these benefits are reaped only after an investment of time and manual intervention that causes many to question the very role of the computer in either process.

In summarizing computer-based lemmatization methods several years ago -- a summary which is equally appropriate to computer-assisted parsing -- Susan Hockey noted that the computer can be employed in three ways (Hockey 1980: 73). Firstly, the computer can use a set of pre-determined algorithms when attempting to establish the form of a word. Secondly, a text can be tagged with this information at the same time as it is input into the computer. Thirdly, the computer can interact with a computer-readable dictionary, retrieve information on a specific word, and then apply it to an electronic text. The advantages of each method, however, are accompanied by limitations.

Inputting *all* the information manually is quite labour-intensive, and to do so is to make only limited use of the capabilities of the computer as a labour-saving device; as well, tagging manually as the text is entered into the computer increases the likelihood of introducing errors and inconsistency to the text. Though the algorithmic approach to determining lemma forms and parts-of-speech is promising, those using this method are generally unable to alter the algorithms and must, then, accept language- and period-specific limitations; algorithms designed specifically for Latin, for example, would not work with German, nor would algorithms for contemporary Spanish completely suit an Old Spanish text. As well, the users of systems which rely on algorithms typically must allow for many exceptions (the most prominent of these being errors resulting from situations not accounted for in the algorithms); hence, the output often requires significant correction. The high level of emendation required in texts processed by such systems once made scholars question the role of the computer as an automator of lemmatization and parsing processes;[6] however, there have been significant advances made in this area.[7]

Employing the computer as a look-up and tagging device in a process which would still involve much manual editing seems time-consuming, especially with the promise of more fully automatic lemmatization and parsing procedures on the horizon. This last scenario, however, is not so bleak, providing that look-up and tagging procedures are efficient, that the system itself -- in terms of computer and human labour -- is non-redundant, and that computer procedures automate all that can be effectively and accurately automated.

2. The *TACT* programs[*]

The *TACT* preprocessing programs reflect consideration of such matters, and incorporate the computer in a partially automated process in which the electronic text is tagged with lemma and part-of-speech/syntactic information simultaneously. Figure 1 outlines the sequence in which the preprocessing programs are used. *PreProc* prepares the initial text file (the input text) for use with the other programs. *MakeDCT* interacts with a master dictionary file containing lemma and part-of-speech forms. *TagText* applies this information to the text[8] and, after one manually edits and corrects the tagged text, *SatDCT* makes a dictionary file which is specific to the input text and both updates existing entries and adds new entries to the master dictionary. Though not a fully automatic system, these programs automate those features which can be accurately carried out by the computer and, together, greatly streamline what can be a very time consuming process.

The first preprocessing program, *PreProc*, works with a tagged text file, as seen in Figure 2,[9] and the information provided by the setup file (*.mks) to produce a set of four output files which are used by the other programs. One of these files is a copy of the input text, which will be eventually tagged with lemma and parsing information; in this file, all tags have been removed and all words which were separated by a continuation character, such as a hyphen, are joined. Another is a list of distinct words which will be used by *MakeDCT* to create a dictionary specific to the input text which, when properly prepared, will be used to tag the text.

Manual interaction occurs in the process at two points: firstly, the text-specific dictionary is edited after using *MakeDCT* and, secondly, the text is proofread after using *TagText*. Proofreading, a stage required even with more automated systems, is necessary to ensure the accuracy of the text. Manually editing the text-specific dictionary file to assign proper lemma and part-of-speech/syntactic forms to the individual words allows the user full control over the representation of that information in the text. This method of editing offers much flexibility, but also requires that one adopt a methodology to guide lemmatization and an accurate grammatical tagset to ensure consistency in the final text.

3. Methodology: Lemmatization

Discussion regarding a lemmatization methodology is warranted, for there has been much recent debate about what specifically the act of lemmatization entails. Though acknowledging that the term itself is ambiguous, in his article "Homography and Lemmatization in Dutch Texts", Boot begins his discussion of lemmatization with the notion that it is "generally defined as the transformation of all inflected word forms contained in a text to their dictionary look-up form" (Boot 1980: 175). To lemmatize a text by this definition, then, is to reduce each word in that text from its inflectional and variant forms to its base form. Questions remain, however, as to whether this theoretical definition of lemmatization is something that can be practicably carried out at the level of application. To illustrate this point, we can turn to Hann who, in his article "Towards an Algorithmic Methodology of Lemmatization," differentiates between a theoretical and a practical approach to the act; lemmatization, he states, can be on one level "the transformation of a corpus of raw textual data to a series of lemmata" or, on another level, "simply the masking of input word-forms . . . in order to treat equivalent forms of the same term as such" (Hann 1975: 140).[10]

Does lemmatization, then, require a full and complete reduction of words to lemmas based on their individual forms and distinct meanings within those forms, or is does it entail only a contraction and masking of words, to what may only *possibly* be their lemma, based on form alone? If each word type could be assigned a unique lemma form, lemmatizing a text would be comparatively uncomplicated; however, because of homography, wherein words which are like one another in form but have distinctly different meanings, this is not so.

In illustration of the larger problem, consider the English word *quail*, which could mean, as a noun, the bird that we eat or, as a verb, to cower. To conflate both words to the same lemma form would be to lose the

distinction in meaning that each separate word has. Improperly treated, this homograph (as others) create ambiguity in the lemmatized text. Such occurrences can be disambiguated by appending lexical information to the lemmatized form. For example, instead of reducing both of the above forms to the lemma *quail*, the two could be represented by the separate lemma forms *quail*<N> and *quail*<V>, where the contents of the angle-brackets denote the part-of-speech for each lemma; when one is parsing as well as lemmatizing, this type of tag can be created automatically by combining elements of each process with the assistance of a word-processor or text editor.

When homographs are of the same part-of-speech, however, the issue becomes more complex. Boot illustrates the problem of homonymy using the word play as it occurs in the two following sentences (Boot 1980: 175):

1. I watched the play.
2. I like fair play.

While tagging for part-of-speech would differentiate between the use of *play* as a noun and as a verb (which is not used in this example), both forms above are nouns, each use carrying with it significant differences in meaning; the first use refers to a drama (*OED*, cf. *play* [III:15]) and the second refers to conduct or dealing (*OED*, cf. *play* [II:12]). Thus, the same graphic form of the same lexical class carries quite differing meanings, and to reduce each to a common lemma would be to ignore their semantic difference.[11]

To be true to the text semantically, then, homographs cannot be simply assigned one lemma or, in complex cases such as that above, one lemma augmented by basic part-of-speech information; rather, each distinct meaning within a common homographic form must be assigned a fully individual lemma to reflect its difference. One solution to this problem is to lemmatize the text following the example of the dictionary;[12] homographs of the same lexical category could be assigned a common lemma stem, but with further tag suffixes to indicate difference. For the above example of the word play, one might tag according to the distinctions offered by the *Oxford English Dictionary* and associate the word *play* in the first occurrence with the lemma *play*<N-III:15> and the second with *play*<N-II:12>. This would require significantly more manual interaction on the part of the person performing the lemmatization than is typical in such projects, but those using lemmatized texts for the purposes of semantic analysis would likely wish distinctions such as these to be noted, if it is at all practical to do so. Those with other concerns may choose not to be so particular.

Because it requires exponentially more effort to remove homographic ambiguity than is necessary to simply mask word forms, lemmatization which fully reflects the semantic difference in homographic forms is infrequently done. Those involved in the creation of the Royal Irish Academy's *Dictionary of Medieval Latin from Celtic Sources* found that "full lemmatization (with removal of ambiguity) . . . demanded more effort on the part of the lexicographer than it was feasible to invest" (Devine 1987: 56; 61). Meunier, who himself has drawn a distinction between graphical and semantic lemmatization, favoured a concentration on graphical forms in his own project (Meunier 1976: 210). Krause and Willée, in their work with German newspaper texts, found that a highly accurate lemmatization, with little ambiguity, came at a cost which exceeded their budget (Krause and Willée 1981: 101). It is no wonder, then, that one criticism of the practice of lemmatization is that it is at most an "operation concealing the semantic problem of homonymy" (Boot 1980: 175). This may very well be the case but, though a lemmatized text may not fully reflect semantic difference in homographic forms, homographs can be disambiguated manually as they arise in data retrieved from that text.

By its nature, then, lemmatization is itself an ambiguous act, and one which significantly transforms a text in ways that can profoundly alter the results of studies performed on it. Because of this, the degree of lemmatization which is chosen for a specific text, and the principles which will be applied in the process of lemmatization, should reflect the intended final use of that text. Figure 3 outlines some general principles of

lemma form reduction in English.[13] In addition to these principles, one should take into account a number of further issues which are commonly considered when reducing a word to its lemma. Are adverbs which are derived from adjectives best lemmatized under the adverbial form, as is common lexicographical practice, or, as Devine has practised, under the adjective (Devine 1987: 58)?[14] Should a gerund functioning as a noun be classified under its original verb form? In some studies it is also useful if words which are derived from the same root-form are lemmatized under the stem of that form; under this principle, the verb or adjective *live* and the noun *life* are conflated to the same lemma (Boot 1980: 175). Furthermore, in many cases it is useful to retain information about the form of the word as it originally appeared in the text, for much data is lost when the past participle of a verb is represented by its infinitive form, or a genitive noun represented by the nominative; for this reason, one should consider the benefits of retaining the original form of a word along with its lemma, and also weigh the advantages of having lexical information available for each word in that text.

4. Methodology: Parsing

Should one choose to parse as well as lemmatize -- actions which are performed at the same time with the preprocessing programs -- some consideration must also be given to the way in which a grammar will be applied to the text. As with lemmatization, one finds a certain range of opinion when considering parsing as an ideal versus parsing as a practical reality; consequently (and also similarly) a range exists among prominent examples of what it means to parse in a scholarly context.

When parsing the *York Computer Inventory of Prose Style* for his study published in *Prose Style and Critical Reading*, Cluett based his system, called the *York Syntactic Code*, upon that laid out in Fries' *Structure of English*. Cluett, while adopting Milic's own revisions of Fries' grammar,[15] made several further alterations to allow him to parse his group of texts in greater detail. His method was to apply a three-digit numeric code to each word to represent its part-of-speech and then to analyze the distribution of those numeric codes (see 16-22); for example, the phrase *after leaving the ship* would appear in the parsed text as the numeric string 513 071 311 011 (Cluett 1976: 19). Ross and Rasche's program, *EYEBALL*, offers a different approach, employing a small built-in dictionary to assign each word only a single letter as a code which represents its lexical category; for example, a noun is assigned *N*, a verb, *V*, an adjective, *J*, and an unknown word, *?*.[16] Those parsing today, however, are not limited in the same way by the technology employed initially by Cluett and Ross roughly twenty years ago -- early studies, for example, were limited to some degree by the technology employed for data storage and retrieval -- so, when deciding upon a tagging methodology to guide parsing, one need not take as minimalist an approach as that taken in *EYEBALL*, nor resort to a numeric system to represent a detailed parsing grammar. Today's technology allows a considerable flexibility but, like the decisions one must make when involved in lemmatizing, when parsing one must decide upon a practical parsing grammar which takes into account the intended use of the text and, of course, one must be prepared for problems arising due to homography.

Ideally, one's target should be [a] a grammar detailed enough to accurately represent the structure of the text and [b] a tagset which reflects an accepted system. Figure 4 outlines a tagset based upon that employed in two recent projects using the preprocessing programs, again for application to English texts.[17] The grammar it reflects is based as much as is practically possible upon that presented by Quirk, and others, in their *Comprehensive Grammar of the English Language* (Quirk 1985).

This tagset, which accounts for word form information alone, attempts to capture the detail of Cluett's system, but represents lexical information in a more visually identifiable form akin to *EYEBALL*'s system; instead of using a numeric system, it employs alphabetic abbreviations of recognizable lexical categories. Thus, the phrase *the dog jumped*, which Cluett would represent as 311 011 021 and *EYEBALL* as *D N V*,

would be tagged as *<DETce-art> <Ns> <LEXV3spa>*. Unlike numeric systems, but like that employed by *EYEBALL*, this method of tagging is more easily and immediately recognizable while proofreading, editing, and analyzing the parsed text.[18]

5. The Dictionary (DCT) file

Ultimately, the principles of lemmatization and the parsing grammar one adopts are reflected in the emendations one makes to the dictionary file; thus, editing the dictionary is a stage central to these processes when using the *TACT* programs. *MakeDCT* has, in the past, retrieved lemma and part-of-speech information for words on which no previous information existed in the master dictionary from the *Oxford Advanced Learner's Dictionary*, an electronic version of which is deposited in the Oxford Text Archive; this option is not currently available, though, and those starting to use the preprocessing programs must build, from the texts being processed, the dictionary from which the computer will retrieve information as the text is parsed and lemmatized. With the master dictionary blank, as will be the case as one begins using the program, the text-specific dictionary will appear as in Figure 5; it is made up of five fields separated by tabs (ASCII 09, represented in the figure by figure of tab). The first field contains the word as it appears in the text and, the second, its part-of-speech; if the word is not found in the master dictionary, *???* is placed in this field. The third and fourth fields contain the lemma form of the word, and the fifth preserves its original, or *raw*, form. The appropriate part-of-speech and lemma form must be added manually to each word in this file with a text editor or a word processor that will handle ASCII text, including extended ASCII characters, without corruption.[19] Using the aforementioned lemmatization principles and parsing tagset, the resultant dictionary file would appear as in Figure 6.[20]

Though one cannot hope at this stage to take into account all lexical categories of a particular form -- for example, *to* may be used as an infinitive marker as well as the preposition it is listed to be (Figure 6) -- errors in tagging are corrected manually as they arise in the tagged text after it is proofread; these emendations are then included in the master dictionary using the program *SatDCT*, and will be given as an option within the parsing tag in the future. Because the results of the manual edit are stored for future use, the master dictionary grows quickly and considerably, thus decreasing the amount of labour necessary to parse and lemmatize in each successive effort. For this reason, those wishing to parse and lemmatize a large text with an empty master dictionary will find it profitable to divide the text into smaller units and complete the process on each smaller text separately, one after another.

Conclusion: Applying the principles

While this process is by no means fully-automatic, the preprocessing programs automate several key and time-consuming parts of the process. My own project, which ultimately has employed stylistic analysis techniques, is based on a text which conflates the four editions of Robert Cawdrey's *A Table Alphabeticall* (Siemens 1994). This required the lemmatization and parsing of some 17,000 words in a file of approximately 125 kilobytes, not including tags. A conservative estimate of the time spent working with the preprocessing programs, excluding that used in determining the principles of lemmatization and a parsing grammar (and tagset), is sixty hours. Some of this time may be attributed to the fact that early modern English is not the fixed system contemporary English is, and variants in spelling had to be entered into the dictionary manually.[21] Those working with other early writing systems may encounter a similar situation; others working in languages with considerable homographic ambiguity, such as Latin or Hebrew, may find that extra time is required for disambiguation.

Once a methodology to guide lemmatization and parsing has been decided upon, the text-specific dictionary edited, the tags applied to the text, the text proofread and corrected, and the master dictionary updated, the

lemmatized and parsed text is ready for analysis; Figure 7 shows an example of this file, the output text. The original, or raw, form of the word is retained with the tag RAW, the results of parsing with the tag POS, and the lemma form of the original word appears without a tag. This file, with the creation of a simple setup file, may then be made into a textual database with *MakeBase* and then can be analyzed by *UseBase* and other *TACT* utility programs. Because *UseBase* automatically concords its textbases, one can locate all forms of a word under its lemma. As well, co-occurrences of lemma forms can be easily retrieved with *UseBase*, and collocations and word distribution patterns mapped. The utility *Collgen*, moreover, will track exact repetitions and generate a rule file whereby these repetitions can be located in the text.

Those wishing to perform similar functions on the parsed text can do so by placing the lemma form within a tag and making the contents of the POS tag the main text, as in Figure 8; this is most easily accomplished with *TagText*, or with the use of the macro language of a word-processor. By revising the initial setup file, a textbase can be created with *MakeBase* in the same way as the file which is used to analyze the lemma forms of a text. Further alterations can be made with a text editor or word processor to take into account punctuation and other structural features of the text which are not captured by parsing and lemmatizing, as seen in Figure 9.

Though *TACT*'s analysis programs offer functions which will assist in most computer-based studies -- those of style and authorship, those involving semantic, morphological, and syntactic analysis, those relying on word indexes and concordances, and others -- files which have been parsed and lemmatized with the preprocessing programs can also be easily modified and exported for use with other packages and on other platforms. As well, the results of studies completed with the analysis programs may be converted into forms which will allow their manipulation and representation by a variety of database, spreadsheet, and other statistical analysis software applications.

Notes

[1] The words *am, are,* and *is*, would appear as *be*, and the words *car, cars, car's* and *cars'* would appear as *car*; the phrase *the boy's cars are different colours* would appear in a lemmatized text as *the boy car be different colour*. In languages other than English, this process would involve similar, though not exactly the same, principles of reduction.

[2] To do the same with an unlemmatized text would still involve lemmatization of a sort, although it would be a "mental lemmatization" in which the user would have to search for each variant form of the lemma each time he or she wished to locate all occurrences of a single word throughout their text (Devine 1987: 56).

[3] Moreover, studies connected with lemmatization and those with parsing are beginning to be considered more closely related than previously thought (see Meunier 1976: 208-9).

[4] For *EYEBALL* and *MicroEYEBALL*, refer to Ross 1981 as well as Ross and Rasche 1972. The *Micro English Parser* was developed by Michael Stairs and Graeme Hirst at the Centre for Computing in the Humanities (University of Toronto). *LexiTex*, an expanded version of *Lemmatiseur* (the Lemmatiser), was developed at the Centre de traitement de l'information (Université Laval); see Lancashire and McCarty 1988 for a description of *Lemmatiseur* (221-2) and, for *LexiTex*, see Lancashire 1991 (505-6). Also to be considered is *THEME*; refer to Bratley and Fortier 1983. For *LEMMA2*, see Klein 1990 as well as Krause and Willée 1981. Laurent Catach's program *GRAPHIST*, to be released in the near future, promises to offer much to those working with French.

[5] Often cited are Ross 1974, Oakman 1975, Cluett 1976, and Milic 1967. As well, there are numerous more

recent works, such as those by Birch 1985, McColly 1987, and others.

[6] For example, refer to Aitken 1971.

[7] Refer to Venezky 1973, Dawson 1974, and Hann 1975, among others. More recent work can be found in Kimmo 1983 and Richie and Russell 1992. Of interest also are current products of this nature, including *PC-KIMMO* (see its review in *Computers and the Humanities* 26.2 [1992]), those produced by Lingsoft (Finland), and IBM's morphological processing program, *OEM*.

[*] Editorial note. Cf. notice on availability of *TACT*.

[8] The preprocessing programs do not, at this time, offer features such as contextual disambiguation; this must be done manually when one proofreads and edits the tagged text.

[9] All examples are from Cawdrey's *A Table Alphabeticall* (Cawdrey 1604).

[10] Meunier, further, distinguishes between the semantic and graphical elements of lemmatization (Meunier 1976: 210).

[11] As a more complex example yet, consider the word *present*. It functions commonly as a noun, in the senses of *gift* (a present) and *now* (the present [time]), as a verb, in the sense of *giving something* (I present this to you), and as an adjective, in the sense of *here* (I am present).

[12] Refer to Krause's discussion of Sture Allén's thoughts regarding the types of information which may be obtained from dictionaries, and how that information can be used to help resolve the problem of ambiguity (Krause and Willée 1981: 102).

[13] This table is adapted from Devine 1980: 58.

[14] For example, is the adverb *quickly* most accurately lemmatized as itself or as the adjective *quick*?

[15] A revised version of Fries' grammar was employed by Milic for his study of Swift's style (Milic 1967).

[16] Refer to Hockey for a descriptive overview of the encoding system used by *EYEBALL* (Hockey 1980: 108-14).

[17] My electronic text of four editions of an early modern dictionary of English, Robert Cawdrey's *A Table Alphabeticall*, is parsed with a similar tagset and lemmatized with a concentration on graphical forms. Ian Lancashire's forthcoming electronic edition of a collection of Elizabethan homilies, *Certaine Sermons or Homilies 1547-1571*, contains tags denoting each word's lemma and part-of-speech based on a similar methodology.

[18] Parsing often includes acknowledgement of syntactic relationships as well; although the tagset in Figure 4 does not account for this, it can be included in the tagset and, thus, applied to the text in the same manner. However, because the preprocessing programs are not context-sensitive in applying tags, tagging at this level can be quite time consuming.

[19] It is crucial to understand that some wordprocessors, such as *WordPerfect*, convert pre-existing input text into a proprietary format, so that if you save an edited file as you normally would, *rather than as an ASCII or DOS file*, it may be unusable by other software. Furthermore, in the case of the tab-character, *WordPerfect* will convert it on entry into a series of space-characters and will not restore these to a tab even if you save the file in DOS format. The best advice is to avoid such a wordprocessor when editing text files and to use

instead a text-editor; a good public-domain editor for DOS, Windows 3.1 and 95, is the *Programmer's File Editor*, available online via the URL http://www.lancs.ac.uk/people/cpaap/pfe/. Otherwise you need to work out a procedure for making sure that the file has not been changed by the wordprocessor. With *WordPerfect*, you will need to reinsert the tab, which is represented as [4,1] in its Typographical Symbols character set.

[20] In this specific case, using a text which is from early modern English, variants in spellings and graphical forms had to be considered as well.

[21] The time was well-spent, however, for the process had the positive effect of normalising the spelling within the text for other types of analysis.

Bibliography

- **AITKEN**, A.J. (1971). "Historical Dictionaries and the Computer", *The Computer in Literary and Linguistic Research: Papers from a Cambridge Symposium* (R.A. Wisbey, ed.), Cambridge: Cambridge University Press: 3-17.
- **BIRCH**, D. (1985). "The Stylistic Analysis of Large Corpora of Literary Texts", *ALLC Journal* 6: 33-8.
- **BOOT**, M. (1980). "Homography and Lemmatization in Dutch Texts.", *ALLC Bulletin* 8: 175-89.
- **BRATLEY**, Paul, and Paul **FORTIER** (1983). "Themes, Statistics, and the French Novel", *Sixth International Conference on Computers and the Humanities* (Sarah K. Burton and Douglas D. Short, eds.), Rockville: Computer Science Press: 18-25.
- **CAWDREY**, Robert (1604). *A Table Alphabeticall of Hard Usual English Words* (Robert A. Peters, Introduction), Gainesville, Florida: Scholar's Facsimiles and Reprints, 1966. Electronic edition edited by R. Siemens. Toronto: Renaissance Electronic Texts, 1994. Electronic texts of all editions (1604, 1609, 1613, 1617) edited by R. Siemens, Oxford Text Archive A-1715-A.
- **CLUETT**, R. (1976). *Prose Style and Critical Reading*, New York: Columbia University Teachers College Press.
- **DAWSON**, J.L. (1974). "Suffix Removal and Word Conflation", *ALLC Bulletin* 2: 33-46.
- **DEVINE**, Kieran, A. **HARVEY**, and F.J. **SMITH** (1987). *The Database of Medieval Latin from Celtic Sources 400-1200: A Study in Computer-Assisted Lexicography*, Dublin: The Royal Irish Academy.
- **FRIES**, Charles C. (1952). *The Structure of English: An Introduction to the Construction of English Sentences*, New York: Harcourt, Brace.
- **GARSIDE**, Roger, G. **LEECH**, and G. **SAMPSON**, eds. (1985). *The Computational Analysis of English: A Corpus-based Approach*, London: Longmans.
- **HANN**, M.L. (1975). "Towards an Algorithmic Methodology of Lemmatization", *ALLC Bulletin* 3: 140-50.
- **HELLBERG**, Staffan (1972). "Computerized Lemmatization Without the Use of a Dictionary: A Case Study from Swedish Lexicography", *Computers and the Humanities* 6: 209-12.
- **HOCKEY**, Susan (1980). *A Guide to Computer Applications in the Humanities*, Baltimore: The Johns Hopkins University Press.
- **KIMMO**, K. (1983). *Two-level Morphology: A General Computational Model for Word-form Recognition and Production*, Helsinki: University of Helsinki, Department of General Linguistics [Publication 11].
- **KLEIN**, Harald (1990). "*INTEXT-PC* -- A Program for the Analysis of Texts", *The New Medium, ALLC-ACH '90 Books of Abstracts and Conference Guide* (Helmut Schanze, ed.), Siegen: 133-6
- **KRAUSE**, Wolfgang, and Gerd **WILLÉE** (1981). "Lemmatizing German Newspaper Texts with the Aid of an Algorithm", *Computers and the Humanities* 15: 101-13.
- **LANCASHIRE**, Ian (1991). *Humanities Computing Yearbook 1989-90*, Oxford: Clarendon Press.

- **LANCASHIRE**, Ian and W. **McCARTY** (1988). *Humanities Computing Yearbook 1988*, Oxford: Clarendon Press.
- **McCOLLY**, W.B. (1987). "Style and Structure in the Middle English Poem *Cleanness*", *Computers and the Humanities* 21: 169-76.
- **MEUNIER**, Jean G., Serge **BOISVERT**, and François M. **DENIS** (1976). "The Lemmatization of Contemporary French.", *The Computer in Literary and Linguistic Studies: Proceedings of the Third International Symposium* (Alan Jones and R.F. Churchhouse, eds.), Cardiff: The University of Wales Press: 208-14.
- **MILIC**, Louis T. (1967). *A Quantitative Approach to the Style of Jonathan Swift*, The Hague: Mouton.
- **OAKMAN**, R. (1975). "Carlyle and the Machine: A Quantitative Analysis of Syntax in Prose Style", *ALLC Bulletin* 3: 100-14.
- **OAKMAN**, R. (1980). *Computer Methods for Literary Research*, Athens, GA: The University of Georgia Press.
- *Oxford Advanced Learner's Dictionary* (1974). Oxford: Oxford University Press.
- *Oxford English Dictionary* (1989). 2nd edn., Oxford: Clarendon Press.
- **QUIRK**, Randolph, *et al*. (1985). *A Comprehensive Grammar of the English Language*, London: Longman.
- **RABEN**, Joseph, and David V. **LIEBERMAN** (1976). "Text Comparison: Principles and a Program", *The Computer in Literary and Linguistic Studies: Proceedings of the Third International Symposium* (Alan Jones and R. F. Churchhouse, eds.), Cardiff: The University of Wales Press: 297-308.
- **RICHIE**, G., G. **RUSSELL**, *et al*. (1992). *Computational Morphology: Practical Mechanisms for the English Lexicon*, Cambridge, MA: MIT Press.
- **ROSS**, Donald Jr. (1974). "An EYEBALL View of Blake's Songs of Innocence and Experience", *Computers in the Humanities* (J.L. Mitchell, ed.), Edinburgh: Edinburgh University Press: 94-108.
- **ROSS**, Donald Jr. (1981). "*EYEBALL* and the Analysis of Literary Style", *Computing in the Humanities* (P.C. Patten and R.A. Holoein, eds.), Lexington, MA: Heath: 85-103.
- **ROSS**, Donald Jr. and Robert **RASCHE** (1972). "*EYEBALL*: A Computer Program for Description of Style", *Computers and the Humanities* 6: 213-21.
- **SIEMENS**, Raymond G. (1994). "The Acorn of the Oak: A Stylistic Approach to Lexicographical Method in Robert Cawdrey's *A Table Alphabeticall*", *Early Dictionary Databases* (Ian Lancashire and T. Russon Wooldridge, eds.), *CCH Working Papers* 4, Toronto: Centre for Computing in the Humanities: 109-21.
- **SPRAYCAR**, Rudy S. (1980). "Automatic Lemmatization in Serbo-Croatian", *ALLC Journal* 1: 55-9.
- **VENEZKY**, R.L. (1973). "Computational Aids to Dictionary Compilation", *A Plan for the Dictionary of Old English* (Roberta Frank and Angus Cameron, eds.), Toronto: University of Toronto Press: 307-27.