Applications of software in the compilation of corpora

Raymond Hickey, Department of English, University of Munich

Abstract

An attempt is made here to sketch some of the applications to which corpus pro- cessing software can be put in the compilation of corpora. The emphasis is on the one hand on the automation of many standard processes, such as text collation and the provision of header information for each file of a corpus, while one the other hand the additional possibilities offered by dedicated corpus software are described. Among the latter special emphasis is put on the transfer of textual data to a database environment for further processing. Further matters such as the use of special fonts for older stages of English and the option of organising the text files of one's corpus for potential users in advance are also discussed.

0 Introduction

Given the nature of the contributions to this volume, the present author thought it appropriate to discuss the uses to which corpus processing software could be put in the compilation and distribution of corpora, especially ones with a diachronic orientation.

Assuming that the compilers of a corpus have reached basic agreement on what periods are to be covered and what texts are to be included, software can be used gainfully from this point onwards. To illustrate possible applications the software system *Lexa* developed by the present author will be used for the ensuing discussion. This programme suite and the three volumes of documentation are available from the Norwegian Computer Centre for the Humanities in Bergen, Norway. Each of the following sections is intended to illustrate a typical situation in which software is useful in the preparatory stage of corpus building. The list is not exhaustive but it does cover the main areas of concern in this phase of text collection and organisation.

1 Text collation

It is safe to assume that more than one individual will be involved in the compilation of a text corpus. Texts will either be scanned or keyed in directly. In either case it is more the exception than the rule to find that a text turns up error-free in the computer. This banal fact increases the status of the individual who is responsible for text correction. Again it is commonplace for more than one version of a text to exist in some intermediary stage of compilation. Sooner or later in such a situation doubts arise as to whether a particular version of a text is the more accurate or the better corrected. The need arises quite quickly for a reliable means of comparing two versions of a single text. Of course the time and date stamp of a file on the operating system level will tell you which of two is the more recent, but age is not a guarantee for correctness.

To resolve this dilemma a programme has been included in the *Lexa* suite which will

compare two files with each other byte for byte and report the differences on screen and write these to a file if required. With *Lexa Compare* one loads two text files from a directory listing and then on pressing a dedicated key, the programme begins a comparison of the two. In the following screen print-outs two versions of the opening line of Chaucer's *Canterbury Tales* are displayed. The version CHAUC_1.TXT contains four errors, *what* is written for *whan* (twice), *kouthe* for *kowthe* and *season* for Chaucer's spelling of *seson*. These errors are highlighted on the screen and the user can immediately recognize which of the versions is the more correct. Errors will be detected anywhere, up to the end of the files chosen. If these are identical you are informed of this.

Less correct version of text

ÚÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³[File 1: CHAUC_1 .TXT 35,071
06/15/93 12:47 Offset: 893]³ ³1 : Whan that Aprill with hise shoures soote
³ ³2 : the droghte of March hath perced to the roote, ³
³3 : And bathed euery veyne in swich licour ³ ³4 : Of which vertu
engendred is the flour; ³ ³5 : What Zephirus eek with his sweete
breath ³ ³6 : Inspired hath in euery hold and heeth
³ ³7 : The tendre croppes; and the yonge sonne ³ ³8 : Hath in the
Ram his half cours yronne; ³ ³9 : And smale foweles maken
melodye, ³ ³10 : That slepen al the nyght with open eye,
³ ³11 : (So priketh hem nature in hir corages,) ³ ³12 :
Thanne longen folk to geen on pilgrimages, ³ ³13 : And Palmeres for to
seken straunge strondes, ³ ³14 : To ferne halwes kouthe in sondry
londes. ³ ³15 : And specially fram euery shires ende
³ ³16 : Of Engelond to Cauntenbury they wende, ³ ³17 : The
hooly blisful martir for to seke, ³ ³18 : That hem hath holpen what
that they were seeke. ³ ³19 : »Bifil than in that season on a day
³ ³20 : In Southwerk at the Tabard as I lay, ³ ³21 :
Redy to wenden on my pilgrymage ³ ³22 : To Caunterbury with
ful deuout corage, ³
³<2222222222222222222222222222222222222222222222222222222222222222222222222222>³ ³ Space: Txt1 - Txt2,
Tab: Split Screen Menu: Shift-Tab ³
ÀÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

More correct version of text

ÚÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³[File 2: CHAUC_2 .TXT 35,071
06/15/93 12:45 Offset: 893]³ ³1 : Whan that Aprill with hise shoures soote
³ ³2 : the droghte of March hath perced to the roote, ³
³3 : And bathed euery veyne in swich licour ³ ³4 : Of which vertu
engendred is the flour; ³ ³5 : Whan Zephirus eek with his sweete

breath ³ ³6 : Inspired hath in euery hold and heeth
³ ³7 : The tendre croppes; and the yonge sonne ³ ³8 : Hath in the Ram his half cours yronne; ³ ³9 : And smale foweles maken melodye, ³ ³10 : That slepen al the nyght with open eye,
³ ³11 : (So priketh hem nature in hir corages,) ³ ³12 : Thanne longen folk to geen on pilgrimages, ³ ³13 : And Palmeres for to seken straunge strondes, ³ ³14 : To ferne halwes kowthe in sondry londes. ³ ³15 : And specially fram euery shires ende
³ ³16 : Of Engelond to Cauntenbury they wende, ³ ³17 : The hooly blisful martir for to seke, ³ ³18 : That hem hath holpen whan that they were seeke. ³ ³19 : »Bifil that in that seson on a day
³ ³20 : In Southwerk at the Tabard as I lay, ³ ³21 : Redy to wenden on my pilgrymage ³ ³22 : To Caunterbury with ful deuout corage, ³
³<22222222222222222222222222222222222222222222222222222222222222222222222222222222>³ ³ Space: Txt1 - Txt2, Tab: Split Screen Menu: Shift-Tab ³
ÀÄÄ ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

This comparison facility does not allow you to alter the contents of a text. Should you wish to check on and edit two texts at once then you can use the similar comparison option in the *Lexa* suite text editor *Lexa Text*.

2 Normalisation

While the critical editions of texts in printed form strive to be accurate in the inclusion of variants, e.g. in the edition of a work attested in different manuscripts, for the electronic form of a text, a normalised version may have very definite advantages in terms of readibility not to say accessibility particularly with older texts or those representing a dialectally divergent language variety.

In essence the process of normalisation consists of replacing variants of a grammatical form by a single form by external consensus, e.g. as the latter is the input to a later standard form or indeed this itself, or by a justifiable decision of the corpus compilers. Despite the almost ideological dislike of normalisation, particularly on the part of medieval scholars, there are obvious advantages to it as it allows later readers to approach a text or texts without undue linguistic difficulty, to see the wood for the trees so to speak.

In contradistinction to printing, in the compilation of corpora no a priori decision has to be made about whether to distribute a normalised text or not. Instead the corpus should include an original unaltered form of a text along with the means for users of the corpus to normalise the text later if they so wish. Just what these means are should be explained briefly.

To begin with recall that the process of normalisation consists of replacing variant forms of a word by some standard or normalised form. What one needs then is software which will recognize every occurrence of a variant form as an instantiation of a normal form. This is realised by creating a list of normal forms and of all forms which represent variants of these. Technically this is achieved by generating a database. For each variant

form there is a single record which at the very least consists of two fields. The first is that for the variant and the second is for the normal form with which the variant is associated. The database will have as many records as there are variant forms to be replaced by normal forms. The extent of the normalisation is thus dependent solely on the number of substitutions which the compiler wishes to carry out. The net result of this procedure is an unaltered and non-normalised version of a file or files along with a database or databases, one per text file, with which the user of a corpus can, if he or she so wishes, generate a normalised version of a text. The programme to use here is called *Database Translate* (DBTRANS) as it translates an input text into an altered output text on the basis of a database. Here is a simply example of how this actually works. A database is created entitled NORMAL.DBF. This contains two fields per record labelled MID_ENG (*Middle English*) and MOD_ENG (*Modern English*) respectively. There are six records with forms of the verb 'have' which occur in Middle English. The programme DBTRANS now examines any input text or texts and if it finds any instances of Middle English forms of 'have' it replaces these by the modern forms specified in the database NORMAL.DBF.

ÚÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³ Lexa utility: DbTrans
(c) Raymond Hickey ³
³ÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ³ ³ Terminology database :
NORMAL.DBF [6] Only words : Yes ³ ³ Input language field : MID_ENG
Open ended : No ³ ³ Output language field : MOD_ENG Ignore
case : No ³ ³ Template for files : NORM_INP.TXT Input files : ASCII
³ ³ Current input file : NORM_INP.TXT Manual oper.: Yes ³ ³ String
to be located : had ³ ³ String to be inserted : HAD
³ ³ No. of replacements : 7 [6] Matching files [1 |
] ³
³ÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ³ ³6 : a simple example. The verb
'HAVE' used to show two forms in the ³ ³13 : HAS, has
³ ³13 : HAS, HAS ³ ³19 : HAVE,
have ³ ³19 : HAVE, HAVE
³ ³22 : HAD, had ÕÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍÍ¸ ³ ³22
: HAD, HAD ³ ³ ³ ³ ³ DbTrans
successfully executed! ³ ³ ³ ³ ³
³ ³ ÔÍÍÍÍÍÍÍÍÍÍÍ Press any key! ÍÍÍÍÍÍÍÍÍÍ¾ ³ ³
³ ³ ³ ³
³ ³ Press <Escape> to abort operation...
30 ³
ÀÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

Needless to say, any normalisation procedure of any reasonable extent will require far more records specifying many more substitutions. This is, however, not a matter of principle but of arranging a suitable database. Again the advantage of normalisation via a translation programme is that the original version of a text is left unimpaired. Furthermore

the normalisation can in fact be carried out by the user of the corpus if he or she so wishes, thus releasing the compiler from the arduous task of generating comprehensive databases for normalisation tasks.

3 Pre-tagging texts

A major decision which the compilers of a corpus have to take is whether the texts of their corpus are to contain any kind of grammatical information. If this decision is made in favour of including such information then a considerable amount of additional work beyond the collection of texts has to be undertaken.

Grammatical information is normally included in corpora by tagging word forms, i.e. by adding a label to words identifying them grammatically. If tagging is to be done then it is only sensible if it is done completely. Quite apart from the actual work of tagging, agreement must be reached in advance on the system of classification to be used. The advantage for users of a corpus is obvious: the retrieval of grammatical information from a text or texts is vastly facilitated if grammatical affiliation has already been specified via tagging.

Given the size of the task, it is imperative to use every resource available for accelerating the process. In effect this means employing tagging software for the purpose. In the *Lexa* suite the main programme, called simply *Lexa*, is designed to tag texts automatically. The operator of the programme, be he or she a compiler or user of a corpus, must specify what words are to be tagged in what way by creating one or more *lemma definition files*. The programme takes note of these definitions and then examines any set of input texts, adding tags to words it deems as representing the grammatical classes for which tags exist in the definition file or files. The programme runs in an automatic, semi-automatic or manual mode. Tagging can be done cumulatively, tags can be exchanged or updated and particularly frequent words (prepositions, articles, etc.) can be excluded from the tagging process.

The result of a tagging operation on two input texts (the opening lines of *Beowulf* and of Chaucer's *Canterbury Tales*) produced the following results. Note that for the purposes of illustration only a selection of tags were specified which means that only a small number of word forms are actually tagged.

Opening of Beowulf in a tagged form

ÚÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³C:\LEXA\DEMO\BEOWULF.LEM
Line 58 Col 1 Page 1 Text 1 ³
³[ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùù
ùù]³ ³Hw+at_INTERROG. ³ ³We Gardena in
geardagum, +teodcyninga, +trym gefrunon, hu_CONJUNC +da ³ ³+a+telingas ellen
fremedon. ³ ³<R 4>
³ ³Oft Scyld Scefing [{scea+tena{] +treatum, monegum m+ag+tum, ³
³meodosetla ofteah, egsode eorlas. ³ ³<R 6>
³ ³Sy+d+dan_CONJUNC +arest [{wear+d{] feasceaft funden,
he +t+as frofre ³ ³gebad, weox under wolcnum, weor+dmyndum +tah, o+d+t+at

³ ³him_INFL_PRO +aghwylc +tara ymbsittendra ofer_LOCATIVE hronrade hyran ³ ³scolde, gomban gyldan. ³ ³<R 11> ³ ³+t+at w+as_BE_PAST god cyning. ³ ³<R 12> ³ ³+d+am eafera w+as_BE_PAST +after_TEMPORAL cenned, geong in geardum, +tone god ³ ³sende folce to frofre; fyren+dearfe ongeat +te hie ³ ³+ar drugon [{aldorlease{] lange hwile. ³ ³<R 16> ³ ³Him_INFL_PRO +t+as liffrea, wuldres wealdend, woroldare forgeaf; ³ ³Beowulf w+as_BE_PAST breme bl+ad wide [{sprang{] , Scyldes ³ ³eafera Scedelandum in. ³ ³<R 20> ³ ³ - - Alt-1 Alt-2 Alt-3 Alt-4 F7=End F10=Save [Sh-Tab=Menus] ³
ÀÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

Opening of the Canterbury Tales in tagged form

ÚÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³C:\LEXA\DEMO\CHAUCER.LEM Line 1 Col 1 Page 1 Text 1 ³
³[ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùùùùùù|ùùù
ùù]³ ³Whan that Aprill with hise shoures_PLURAL soote ³ ³the_ART droghte of_PREP March hath_HAVE perced to the_ART roote, ³ ³And bathed euery_IND_PRO veyne in_PREP swich licour ³ ³Of_PREP which vertu engendred is the_ART flour; ³ ³What Zephirus eek with his_POSS_PRO sweete breath ³ ³Inspired hath_HAVE in_PREP euery_IND_PRO hold and heeth ³ ³The_ART tendre croppes_PLURAL; and the_ART yonge_PAST_PART sonne ³ ³Hath_HAVE in_PREP the_ART Ram his_POSS_PRO half cours yronne_PAST_PART; ³ ³And smale foweles_PLURAL maken melodye, ³ ³That slepen al the_ART nyght_X_PHON with open eye, ³ ³(So priketh hem nature in_PREP hir_POSS_PRO corages_PLURAL,) ³ ³Thanne longen folk to geen on_PREP pilgrimages_PLURAL, ³ ³And Palmeres_PLURAL for_PREP to seken straunge strondes_PLURAL, ³ ³To ferne halwes_PLURAL kowthe in_PREP sondry londes_PLURAL. ³ ³And specially fram euery_IND_PRO shires_PLURAL ende ³ ³Of_PREP Engelond to Cauntenbury they wende, ³ ³The_ART hooly blisful martir for_PREP to seke, ³ ³That hem hath_HAVE holpen what that they were_COPULA seeke. ³ ³ Bifil that in_PREP that seson on_PREP a_ART day ³ ³In_PREP Southwerk at the_ART Tabard as I lay, ³ ³Redy to wenden on_PREP my_POSS_PRO pilgrymage_ROMANCE ³ ³To Caunterbury with ful deuout corage_ROMANCE, ³ ³ - - Alt-1 Alt-2 Alt-3 Alt-4 F7=End F10=Save [Sh-Tab=Menus] ³
ÀÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

Tagging texts before their distribution is something which later users may view as a

linguistic straightjacket as it imposes the grammatical classification scheme of the compilers on the user. Seeing that there is tagging software available, many compilers may now prefer to leave this work to the corpus users, or to some sub-group, such as researchers in another university who would be prepared to carry out this task. As universities have to economise on resources, tagging by the compilers is likely to become less likely in future, especially as partial tagging is not viewed as a sensible course of action. You either tag completely or not at all. If you decide to do so you may bind your capacities in a manner which you come to regret later.

This would appear at least to be the case for major projects like the Helsinki corpus. With the arrival of smaller more specialised corpora, tagging may become feasible particularly if it is directly connected with the research interests of the corpus compilers.

4 Using Cocoa headers

Independent of the question of whether to tag or not to tag, compilers of a corpus should consider whether it would be of avail to future users to include some information on the nature of the texts before distributing these. This decision has fortunately been made in favour of supplying such information by the compilers of the Helsinki corpus. The format they have chosen for the inclusion of text-relevant information is what is commonly known as the Cocoa header. Note that header information is placed at the top of a file and has nothing to do with grammatical classifications included in the body of a text.

Parameters of the Cocoa header

1: <B = 'name of text file'>	2: <Q = 'text identifier'>
3: <N = 'name of text'>	4: <A = 'author'>
5: <C = 'part of corpus'>	6: <O = 'date of original'>
7: <M = 'date of manuscript'>	8: <K = 'contemporaneity'>
9: <D = 'dialect'>	10: <V = 'verse' or 'prose'>
11: <T = 'text type'>	12: <G = 'relation to foreign original'>
13: <F = 'foreign original'>	14: <W = 'relation to spoken language'>
15: <X = 'sex of author'>	16: <Y = 'age of author'>
17: <H = 'social rank of author'>	18: <U = 'audience description'>
19: <E = 'participant relation'>	20: <J = 'interaction'>
21: <I = 'setting'>	22: <Z = 'prototypical text category'>
23: <S = 'sample'>	24: <P = 'page'>
25: <L = 'line'>	26: <R = 'record'>

Although providing headers is not comparable to the task of tagging a corpus, it nonetheless requires an additional amount of work to specify the values for these parameters for the texts of a corpus. The advantages, however, are considerable.

With the *Lexa* suite the contents of a Cocoa header can be accessed by the information retrieval software. This is realised as follows: a programme (called *Cocoa*) extracts the header information from any set of input files and deposits this in a database. Then with the database manager of the suite (called *DbStat*) one can load the database just created and impose a filter on it. By this is meant that only those records remain visible which match a

certain user-specified condition.

Assuming that one generates a database of the Cocoa header information in the files of the Helsinki corpus and loads the database manager then one could specify a filter to which only those records (i.e. file headers in database form) correspond which represent translations (Item 13) of Middle English (Item 6) prose (Item 10) texts. A list of the files for which this header information obtains can be generated by creating a list from the field information for Item 1 (*name of text file*). The list file created by these steps can in its turn be used as the source of the file names for an information retrieval operation with other parts of the *Lexa* suite so that only Middle English prose translations from the corpus are examined. In addition the user can specify with the retrieval programmes from the set (such as the pattern matcher *Lexa Pat* and the programme for locating syntactic contexts *Lexa Context*) that the Cocoa information of the files examined be enclosed in the output file of statistics generated during a search.

The example just given is typical inasmuch as it illustrates how different parts of the *Lexa* suite link up together. For any prospective operators of the programme package, be they corpus compilers or users, it is essential to grasp the inter- relationships between items of software.

5 Word indexes

Among the simplest of tasks to carry out with any corpus processing software is the production of lists of unique words from source text files. Despite this obvious simplicity this type of additional file is very commonly demanded by linguists examining a corpus. To this end the programme *Lexa Words* in the *Lexa* suite has been created. The programme takes any text file or set of files and generates a list of all the words which occur uniquely in the input. Once this list has been created the user can consult it via a pop-up window in which a sorted list appears together with the frequency of the words noted when generating the list in the first place.

ÚÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³ BEOWULF.TXT
Offset: 1,342 ³ ³ ³37 : Hw+at_INTERROG.
²³ ³38 : We Gardena in geardagum, +teodcyninga, +trym gefrunon, hu_CONJUNC +da
²³ ³39 : +a+telingas ellen fremedon. ÚÄÄÄÄÄÄÄÄÄ [145 of 280] ÄÄÄÄÄÄÄ ²³
³40 : <R 4> ³ heold ³ 1 ² ²³ ³41 : Oft Scyld Scefing
[{scea+tena{]³ Heorogar ³ 1 ² ²³ ³42 : meodosetla ofteah, egsode eorla³
heresped ³ 1 ² ²³ ³43 : <R 6> ³ Hi ³ 2 ² ²³
³44 : Sy+d+dan_CONJUNC +arest [{wear+³ hie ³ 2 ² ²³ ³45 : gebad,
weox under wolcnum, weor³ hildew+apnum ³ 1 ² ²³ ³46 : him_INFL_PRO
+aghwylc +tara ymb³ him ³ 11 ² ²³ ³47 : scolde, gomban gyldan. ³
hine ³ 3 ² ²³ ³48 : <R 11> ³ his ³ 1 ² ²³
³49 : +t+at w+as_BE_PAST god cyning. ³ hl+aste ³ 1 ² ²³ ³50 : <R 12>
³ holm ³ 1 ² ²³ ³51 : +d+am eafera w+as_BE_PAST +afte³
hringedstefna ³ 1 ² ²³ ³52 : sende folce to frofre; fyren+de³ Hro+dgar
³ 1 ² ²³ ³53 : +ar drugon [{aldorlease{] lange³ Hro+dgare ³ 1 ² ²³ ³54 :
<R 16> ³ hronrade ³ 1 ² ²³ ³55 : Him_INFL_PRO +t+as

liffrea, wul³ hu ³ 1 ² ²³ ³56 : Beowulf w+as_BE_PAST breme
bl+aÀÄÄÄÄÄ [Search: Ù; Abort: Esc] ÄÄÄÄ ²³ ³57 : eafera Scedelandum in.
²³ ³58 : <R 20> ²³ ³59 :
Swa_CONJUNC sceal [{geong{] [{guma{] gode gewyrcean, fromum feohgiftum ²³ ³ Index
File: BEOWULF.WDX ³
ÀÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

This type of word index could be created in advance by the compilers of a corpus and supplied on the distribution medium, thus obviating the need for users to generate such lists themselves. Given the greatly increased storage capacity of mediums such as CD-ROM disks the additional space required for such index files should not be a deterrent to offering them with the primary text files of a corpus.

6 Lexical databases

Closely related to word indexes is another type of file which is useful in analysing texts lexically. This is what is termed a lexical database. Recall that a database is a type of file in which information in stored in table-like form. The rows of the table contain different fields and the columns the contents of these fields. With the main programme *Lexa* of the suite under discussion it is possible to derive a database from any input text or texts. By this is meant that you load or specify an input text and then demand of the programme that it extract information on each word, storing this in a record with four fields. The first contains the word form itself, i.e. the token which is found. The second contains the tag, if any, which has been associated with the word form in question. For the third field the frequency of the word in the input text has been noted. The frequency is stored cumulatively which means that if you run the lexical database function on a series of texts, the entry for FREQUENCY is incremented for every find of a particular word form in each text. Thus if you, for example, divided *Beowulf* into six texts and generated a lexical database for each text, using the same output database, then the frequency field for any given form would contain the total number of occurrences in all the six texts taken together. In the fourth field the word is deposited in reverse order of spelling. The idea behind this is to allow the creation of reverse order dictionaries which would give information on grammatical endings. If you sort a lexical database alphabetically going on the field REVERSE then you end up with the records ordered according to the end and not the beginning of the words in the field TOKEN.

ÚÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³BEOWULF.DBF
DB: 1 Field: 1 Col: 1 Rec: 119 [272]³
³<2222222222222222222222222222222222222222222222222222222222222222222222222>[|]³ ³TOKEN C 32 hine
³ ³LEMMA C 16 INFL_PRO
³ ³FREQUENCY N 6 3 ³ ³REVERSE C
32 enih ³ ³
³ ³ ³ ³
³ ³ [-]PgUp [-]PgDn -G=Goto F1=Help Alt-F7=Desktop F10=Browse

[Shift-Tab=Menus]³
ÀÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

Whether compilers of a corpus would feel like including lexical databases for their primary text files depends on the status of lexical analysis on their own research horizon. Once more the question of the space which such files would occupy diminishes when one considers the large capacity of storage mediums nowadays.

7 Concordance files

Looking at words in isolation is one aspect of lexical analysis. Another which stresses contextualization is the viewing of words in context. Here one is dealing with a file type which while not very sophisticated from a computing point of view nonetheless has its justification is providing valuable information to users of a corpus.

The main programme *Lexa* can once more be employed to generate the type of file in question. There are basically two types of concordance, one in which the keyword being considered is displayed in the centre of a text line with a certain number of words to the left and right of it included as context; this is known as a KWIC, or 'keyword in context' file.

Keyword in context, KWIC-file

ÚÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿
³33úúú:úúúAndúmadeúforwardúerlyúforúútoúúryse, ³
³34úúú:úúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúToúútakeúoureúweyútherúasúIúyowúdeuyse.
³ ³37úúú:úúúúMeúthynkethúitúacordauntúútoúúresoun ³
³38úúú:úúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúToúútelleúyowúalútheúcondicioun ³ ³
³ ³[úWelú]
³ ³ ³
³24úúú:úúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúWelúúnyneúandútwentyúinúaúcompaignye
³ ³29úúú:úúúúúúúúúúúúúúúúúúúúúúúúúúAndúúwelúúweúwerenúesedúatteúbeste.
³ ³ ³ ³[úWhanú]
³ ³ ³
³1úúúú:úúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúWhanúúthatúAprillúwithúhiseúshouresúsoot
e ³ ³30úúú:úúúúúúúúúúúúúúúAndúshortly,úúwhanúútheúsonneúwasútoúreste, ³
³ ³ ³[úWhatú]
³ ³ ³
³5úúúú:úúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúúWhatúúZephirusúeekúwithúhisúsweeteúbrea
th ³ ³18úúú:úúúúúúúThatúhemúhathúholpenúúwhatúúthatútheyúwereúseeke. ³
³40úúú:úAnúwhicheútheyúwere,úandúofúúwhatúúdegree, ³
³41úúú:úúúúúúúúúúúúúúúúúúúAnúeekúinúúwhatúúarrayúthatútheyúwereúInne; ³
ÀÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

The second major type has the keyword in a separate column to the left with the text line from which it is taken following it. Here one is dealing with a KWOC or 'keyword out of context' file. Both file types can be generated quite easily with *Lexa*. Furthermore the information of a concordance file can be transferred to a database environment to enable users to avail of the additional manipulative power of the latter type of file.

Keyword out of context, KWOC-file

ÚÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ³33 : And made forward erly for
{to} ryse, ³ ³34 : {To} take oure wey ther as I yow deuyse.
³ ³37 : Me thynketh it acordaunt {to} resoun ³ ³38 :
{To} telle yow al the condicioun ³ ³
³ ³[Wel] ³ ³
³ ³24 : {Wel} nyne and twenty in a compaignye
³ ³29 : And {wel} we weren esed atte beste. ³ ³
³ ³[Whan]
³ ³ ³ ³1 : {Whan} that
Aprill with hise shoures soote ³ ³30 : And shortly, {whan} the sonne
was to reste, ³ ³ ³ ³[
What] ³ ³
³ ³5 : {What} Zephirus eek with his sweete breath ³ ³18
: That hem hath holpen {what} that they were seeke. ³ ³40 : An whiche they
were, and of {what} degree, ³ ³41 : An eek in {what} array that they
were Inne; ³
ÀÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÙ

8 Font considerations

The corpora and corpus projects under construction which are presented in the present volume all refer to diachronic English. If the time span in a particular instance stretches back far enough then texts will involve special characters for Middle and Old English. The practice with historical corpora has been to represent special symbols of historical stages of the language by using so-called escape sequences. For instance the Old English character *thorn* is represented by '+t' in the Helsinki corpus, the *eth* symbol is indicated by '+d' and so forth. This encoding has the advantage of portability. The corpus texts only include characters with numeric values between 32 and 126 in the ASCII set and are transferrable to and readable on computer systems operating on a so-called 7-bit basis. The obvious disadvantage is that readability drops drastically with older texts. Something like *Beowulf* is undecipherable in the ‘escape-sequence’-form.

A practical solution to this problem, presented in the *Lexa* suite, is to use a supplied programme to convert the sequences to single characters with the correct shapes so that an Old English text on screen looks more or less identical to one in printed form.

The scheme devised by the present author utilises the ability of personal computers with colour monitors to display characters with customized shapes on screen. The programme which makes the alterations inserts the redefined symbols of the screen (which

have the shapes of the Old and Middle English characters) in any set of input texts at those points where it encounters an escape sequence, e.g. it inserts the *yogh* symbol when it hits on '+g'. The conversion is reversible so that texts can be restored to their original form if desired. The numerical values of the redefined characters with Old and Middle English shapes are as following.

Escape sequence	Actual character	Letter name	ASCII numerical value for redefinition by Lexa programme Make Symbols
+a	æ	'ash',l.c.	145 +A
	Æ	'Ash',u.c.	146 +d
		'eth',l.c.	253 +D
	'Eth',u.c	252 +g	'yogh',l.c.
		243 +G	'Yogh',u.c.
242 +t	"	'thorn',l.c.	245 +T
	"	'Thorn',u.c.	244
+tt	ø	'crossed thorn'	248 +TT —
		'crossed Thorn'	246
+e	É	'e caudata'	144 +l œ
		'pound sign'	156

Taking a typical text such as *Beowulf* and loading it with one's text editor or word processor leads to one being presented with a text which is convenient for computer manipulation but hardly readable to the Old English scholar.

Beginning of Beowulf with 'escape-sequence' coding

[} [\BEOWULF\] }] <R 1> Hw+at. We Gardena in geardagum, +teodcyninga, +trym gefrunon, hu +da +a+telingas ellen fremedon. <R 4> Oft Scyld Scefing [{scea+tena{] +treatum, monegum m+ag+tum, meodosetla ofteah, egsode eorlas. <R 6> Sy+d+dan +arest [{wear+d{] feasceaft funden, he +t+as frofre gebad, weox under wolcnum, weor+dmyndum +tah, o+d+t+at him +aghwylc +tara ymbsittendra ofer hronrade hyran scolde, gomban gyldan. <R 11> +t+at w+as god cyning. <R 12> +d+am eafera w+as +after cenned, geong in geardum, +tone god sende folce to frofre; fyren+dearfe ongeat +te hie +ar drugon [{aldorlease{] lange hwile. <R 16> Him +t+as liffrea, wuldres wealdend, woroldare forgeaf; Beowulf w+as breme bl+ad wide [{sprang{] , Scyldes eafera Scedelandum in.

If one takes this text, however, and runs it through the programme *Make Symbols* which is supplied in the *Lexa* suite then a number of substitutions are made and certain high ASCII characters are inserted where escape sequences were found in an input text. Now under the important assumption that (i) you are using a computer with a colour monitor, typically a VGA video adapter based system, and (ii) that you have loaded the supplied Old / Middle English font of the *Lexa* suite then the stretch of text printed above should now look like the following.

Beginning of Beowulf with Old English characters [} [\BEOWULF\] }] <R 1> Hwæt. We Gardena in geardagum, "eodcyninga, "rym gefrunon, hu a æ"elingas ellen fremedon. <R 4> Oft Scyld Scefing [{scea"ena{] "reatum, monegum mæg"um, meodosetla ofteah, egsode eorlas. <R 6> Syan ærest [{wear{] feasceaft funden, he "æs frofre gebad, weox under wolcnum, weormyndum "ah, o"æt him æghwylc "ara ymbsittendra ofer hronrade hyran scolde, gomban gyldan. <R 11> "æt wæs god cyning. <R 12> æm eafera wæs æfter cenned, geong in geardum, "one god sende folce to frofre; fyrenearfe ongeat "e hie ær drugon [{aldorlease{] lange hwile. <R 16> Him "æs liffrea, wuldres wealdend, woroldare forgeaf; Beowulf wæs breme blæd wide [{sprang{] , Scyldes eafera Scedelandum in.

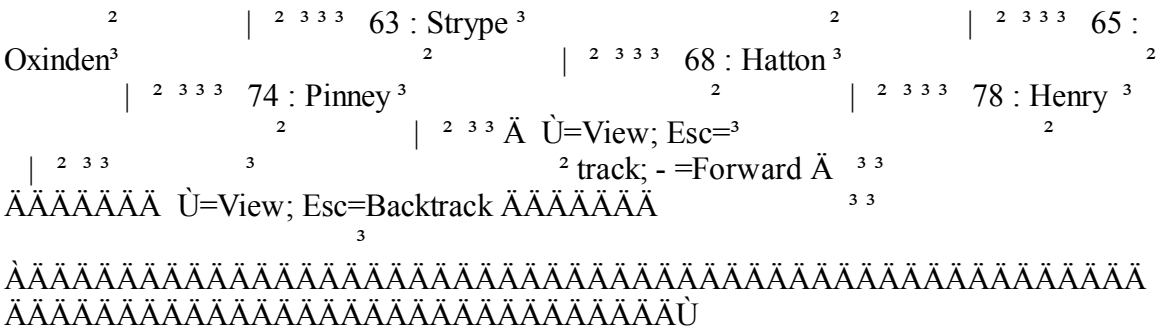
In the interests of a unified system, prospective compilers of diachronic corpora are advised to keep the encoding system used by the Helsinki corpus for special characters. If they do then the software already available for the latter corpus can be used without any alteration in a newer corpus which complies to the original codification scheme. Users of several corpora will only need one special video font, namely that supplied with the *Lexa* suite and can then view any text from a selection of corpora without further system adjustment. Note this coding scheme is also that used by the present author for the medieval texts in the *Corpus of Irish English* (see elsewhere in this volume for details).

8 Organisational considerations

Once a corpus has been completed a practical question arises for its potential users. How do they gain an overview of just what the corpus contains? The simple answer with many corpora is to consult the manual. However, given that a corpus is an electronic library of texts, it is surely natural to expect that this question can be answered electronically. Not only that, there are bound to be additional advantages to be accrued from looking at the contents of a corpus on the computer rather than in printed form. To this end the programme *Corpus Manager* has been designed and included in the *Lexa* suite.

Basically what the corpus manager does it to provide one with a table of contents for a corpus with up to three levels of depth. There is a contents window for each level and the user can choose any text and view it by selecting it in the current window.

ÚÄÄÄ ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ¿ ^{3} [F7=Desktop; F9=Settings] $^{3\ 3}$ ÄÄÄÄÄÄÄ Level 1 [4 of 19] ÄÄÄÄÄÄ $^{3\ 3\ 3}$ 1 : Overview; Cocoa info | ^{2} $^{3\ 3\ 3}$ 2 : Beaumont | ÄÄÄÄÄÄÄ Level 2 [1 of 2] ÄÄÄÄÄÄ $^{3\ 3\ 3}$ 4 : Plumpton | ^{3} 13 : Thomas More | $^{2\ 3\ 3\ 3}$ 12 : More et al. | ^{3} 17 : Margaret Roper | $^{2\ 3\ 3\ 3}$ 19 : Cromwel ÄÄÄÄÄÄÄ Level 3 [1 of 3] ÄÄÄÄÄÄ | $^{2\ 3\ 3\ 3}$ 21 : Cumberl^{3} 14 : Letter to wife ^{2} | $^{2\ 3\ 3\ 3}$ 26 : Knyvett^{3} 15 : Letter to daughter ^{2} | $^{2\ 3\ 3\ 3}$ 28 : Harley ^{3} 16 : Letters to M.Roper ^{2} | $^{2\ 3\ 3\ 3}$ 30 : Paston ^{3} ^{2} | $^{2\ 3\ 3\ 3}$ 33 : Ferrar ^{3} ^{2} | $^{2\ 3\ 3\ 3}$ 36 : Barring^{3} ^{2} | $^{2\ 3\ 3\ 3}$ 45 : Proud e^{3} ^{2} | $^{2\ 3\ 3\ 3}$ 56 : Gawdy ^{3} ^{2} | $^{2\ 3\ 3\ 3}$ 58 : Haddock^{3}



Furthermore online searching of texts is possible. Searching can be on a global level encompassing all texts of a corpus and including unspecified elements in search strings by the use of wild cards. Texts can be extracted from a corpus and printed separately if required.

The corpus manager is particularly suitable for those corpora which consist of many small parts and where there is a hierarchical relation between these. Technically to adapt a corpus for use with the programme *Corpus Manager* one must create a single file, then place level markers at strategic points in this file indicating where the breaks are in the text so to speak. A special text editor will carry out the task of marker placement quite easily. The programme then indexes the file which has been prepared in this fashion. Once this has been done, one can consult the corpus, now in the form of a text database, at will. There is no restriction on the number of text databases, so that one could conceivably divide a corpus into several blocks, each with an internal structure determined by the compilers of the corpus. To illustrate this technique there is a text database of letters from the history of English included as part of the *Lexa* suite.

9 Conclusion

The present sketch is intended to offer the interest linguist engaged in corpus compilation an idea of what can be done to both facilitate the process of compilation for him/herself and to increase the gainful use to which the corpus can then be put once it is completed. In all instances, the programmes of the *Lexa* suite do not have to be altered in any major way, apart from the operator of the programmes creating his/her own configuration for in some instances (this is standard procedure). The net result is a greater degree of automation for many processes which represents a saving in resources which renders in turn many a task more feasible and brings forward the distribution date for many an interesting corpus.

References

Hickey, Raymond 1993a. *Lexa. Corpus Processing Software, 3 Vols. Vol.1: Lexical Analysis. Vol.2: Database and Corpus Management. Vol.3: Utility Library* Bergen: Norwegian Computing Centre for the Humanities.

Hickey, Raymond. 1993b. "Corpus data processing with Lexa", *ICAME Journal* 17: 73-96.

Hockey, Susan and Ian Marriott 1980. *Oxford Concordance Program: Users' manual*.

Oxford: Oxford University Computing Service.
Johansson, Stig 1986. *The tagged LOB Corpus: User's manual*. Bergen: Norwegian Computing Centre for the Humanities.
Johansson, Stig and Anna-Brita Stenström (eds.). 1991. *English computer corpora: Selected papers and research guide*. Berlin: Mouton de Gruyter.
Kytö, Merja, Ossi Ihalainen, and Matti Rissanen (eds.) 1988. *Corpus linguistics hard and soft*. Amsterdam: Rodopi.
Kytö, Merja 1991. *Manual to the diachronic part of the Helsinki corpus of English texts*. Helsinki: Department of English.
Kytö, Merja and Matti Rissanen 1988. The Helsinki Corpus of English Texts: Classifying and coding the diachronic part. In *Corpus linguistics*, ed. by M.Kytö et al. 169-180. Amsterdam: Rodopi.