A Database Package for Lexicological Work on Personal Computers

8th. International Conference on English Language Research on Computerized Corpora

Helsinki, 21-24 May 1987

Raymond Hickey, Universität Bonn

Abstract

The purpose of the present paper is to provide an overview of the package LEXA, a set of databases for lexicological work on personal computers. After preliminary remarks on types of computers a detailed description of the individual sections of LEXA is offered. The indexing and pattern matching facilities are dealt with and the possibilities of creating and working with a lexicological database within this environment are elaborated on. A test case of a Middle English text is discussed and typical operations to be performed on it are presented as examples of a concrete application of LEXA.

0. Introduction

The purpose of the LEXA database package is to put at the linguist’s disposal a series of tools for carrying out lexicological work on personal computers. Before starting to describe this package in detail allow me to make a few remarks on philosophy. The database package LEXA is part of a larger project which I have termed A Computer Package for Linguists. This is series of facilities for linguists using MS-DOS personal computers and can be divided into three main parts. The main emphasis is on foreign language fonts and on databases which are of relevance to linguistic work. The first part is PolyFont which contains some 30 special fonts for use with all personal computers and 24 matrix needle printers. The second part is DataSet which is a collection of four databases which are also intended for use on personal computers. These databases are Litera, a bibliographical management system, Contacts an address management system for university departments and publishers, Studia, a student record management system and Addressa, a management system for normal addresses.

The third part of the package is LEXA, a lexical data management system for linguist and the object of interest in the present lecture.

In addition, the package contains miscellaneous items such as a file manager, macros and templates for commercially available personal computer software such as the word processor WordPerfect or the integrated programme Framework. A Computer Package for Linguists comes with a handbook of some 200 pages which explains the use of the foreign language fonts, the databases and the miscellaneous files. It is public domain software and can be had from the present speaker at the cost of the materials; distribution is intended to begin in the autumn of this year.
0.1. Clearing the ground: The possibilities of personal computers

The choice of type of computer of which to design *A Computer Package for Linguists* with is a deliberate one. There are many advantages to be accrued from doing one’s work as a linguist on a personal computer as opposed, for example, to availing of mainframe facilities in the computer centre of one’s university. The single greatest advantage is that one can control directly the output of processing. This is of particular relevance when working with texts especially when these involve non-standard symbols such as a phonetic alphabet or the special letters of a previous stage of a language (e.g. Old English). Other advantages such as being independent of waiting lists or of possible breakdowns at the mainframe centre are obvious.

Up to a few years ago personal computers did not simply have the speed of processing or the storage capacity typical of mainframes, let alone sophisticated software for special purposes. With the advent of faster personal computers based on the Intel 80286 processor and with the use of high performance Winchester disks and not least with the development of professional software for personal computers one can now safely say that they have come of age. This statement is even more true when one considers the new generation of machines based on the Intel 80386 processor which will be widely available in the next year or so.

For the linguist involved in lexicological work the developments just outlined are of direct benefit. It is now possible to have database management systems capable of handling large amounts of data easily and speedily. For example, there is no difficulty in handling a bibliographical database with 10,000 records or, say, a text of 500 pages on a powerful type of personal computer. The database package to be described here is intended for amounts of data of this order so that the linguist no longer has to resort to a mainframe, at least not for reasons of handling capacity.

1. An outline of LEXA

The database package *LEXA* consists of three subparts (i) KWIC - KWOC, (ii) CORPUS and (iii) PAT - MAT. Each of these subparts fulfills certain specific functions. However, all three are intended to be used in combination for the purposes of lexicological analysis.

To illustrate the type of analysis which one can carry out with the *LEXA* package allow me to present a hypothetical case. This is also the test case used in the documentation which I supply with *LEXA*.

The starting point of my hypothetical case is the desire to analyse lexicologically the Prologue to Chaucer’s *Cantebury Tales* and the first of the tales, *The Knight’s Tale*. This is a fair-sized text of some 3,100 lines and can be used to extrapolate processing times for larger texts should these be present.

The very first step is of course entering one’s text. This can be done manually by typing it or, if the necessary facilities are available, by means of an optical character recognition device. The net result must in any case be a simple ASCII file with no formatting features specific to any particular word processor. The text can, however, contain punctuation, tab stops, inverted commas, etc. When serving as input to *LEXA* these are be removed. This is achieved with a facility provided internally in *LEXA* which strips any input text of its punctuation.
Note that the starting point is a text. However, all processing within the LEXA package is done on the basis of primary data being available in database form. This means that the input text must be converted into the records of a database. The conversion is effected quite simply within LEXA by reading the input text into an empty database line by line. The result is a database in which each record contains a single line. Each record can of course contain additional fields with specific information. An item of information which it is desirable to include is the number of the line. Each line of the input text can then be accessed via the field with the line number. It is possible to treat the resulting database as if it were a page-oriented text by specifying a length for a page (say 56 lines) and defining a field “page number” in each record. This field then begins at “1” and is incremented by “1” every 56 records. An advantage to this is that one can produce an index with the database which is exactly parallel to the text in its form as a text.

The line-oriented database which is the result of the first step in LEXA (after removal of punctuation) now forms the object for the actual operations possible with LEXA. Here it is necessary to consider the subparts of the package individually.

1.1. KWICKWOC

The first part, Kwic-Kwoc, allows a further differentiation of the line-oriented database. Basically one can perform one of two operations. On the one hand one can produce a KWIC index, i.e. with the keyword in context, on the basis of the initial database. Each record of the database is read into a further database in which line delimiters are added and each word of the line is tagged with a symbol such as an asterisk. The user of the package has the chance within Kwic-Kwoc to specify the type of line delimiters he wishes to have and the type of word tag for a Kwic or a Kwoc.

The second basic operation which one can perform with Kwic-Kwoc is of course to produce a KWOC index, i.e. with the keyword out of context. Here one can choose from a series of options. Basically what happens is this. From each record of the line-oriented database a word is removed one at a time until the given line is exhausted and the index programme moves to the next record to repeat the procedure. This means that one can simply change a text into a word list. In doing this one can specify that the new database containing the word list is to include the original line in its entirety, or not. One can also specify that each word is to form a record of its own or that all the words of a single line of the input database are to be grouped into a single record.

Within Kwic-Kwoc it is also possible to determine the longest line in a database, the number of words in the longest line, the number of words in the input text, etc. When creating a line-oriented database one can choose to include only the beginnings of lines or only the end of lines if one is interested in metrical of poetic texts, for example.

The Kwoc index produced from the line-oriented database contains numbering of the words taken from the database. The line number is also adopted from the input database. Thus any given word is uniquely indentifiable vis … vis the original text via its line and word number. In the resulting database each record contains at least three items of information, the word itself, the number of the line
it came and its numerical position as a word in the line, e.g. “hadde”, “156”, “4” means that this word is the fourth word on the 156th line of the input text.

As the entire LEXA package is designed to be as flexible and open as possible the user has a series of choices with regard to the manner in which various options are realized. Thus he can, for example, with a Kwoc index specify that only the word count is to be done by the index programme, and that the line numbering is to be adopted from the input database. This is called for where one has an input text with irregular line numbering. In such a case the user must provide the line numbering himself, the word numbering being done by the index programme.

1.2. PATMAT

Once one has a text in the form of a line-oriented database one can perform a variety of further operations which involve statistics in the widest sense. These functions are provided by another part of the LEXA package, Pat-Mat. This is intended to be used for pattern matching operations. The basic idea is to attempt, within this database environment, to emulate certain features of the SNOBOL4 programming language (a language explicitly designed to carry out string manipulation operations).

When one loads Pat-Mat one is prompted to choose a database for the operations which are to be carried out. In the case at hand the line-oriented database could be specified here. All the pattern searches done by Pat-Mat can be echoed to the printer and/or exported to a text file. This means that the result of a search is written as a text which can then be examined by the user and retained as textual data for some later purpose. There are five main types of pattern search available in Pat-Mat. These are as follows:

1) Search for a string (non-specific)
2) Search for a word
3) Find a word if beginning of a line
4) Find a word if end of a line
5) Search for a string which begins in a certain way and ends another way
6) Search for a word with a specified preceding word and a specified following word

Some of these types may be known to users of SNOBOL4 or the SPITBOL version of this programming language. Thus option 3 (word at the beginning of a line) corresponds to the anchored mode of SNOBOL4.

With each of these options the user can choose to have the line numbers of the finds or to have a frequency count of the finds, i.e. if a string occurs ten times in a text then the result of the search can either specify the lines where the form occurs or simply the number of times it occurs (or of course you can specify that both types of information should be given in the resulting text file).

The first option is intended to comb through the specified database for any matches to a given string. Here it is of no importance how the string relates to the beginning or end of a word or line.

The second option carries out a search for a string which must at the same time be a whole word.
The third and fourth options are sensitive to the position of the word (or string) in a line, i.e. it must occur at the beginning or end of a line respectively. The advantages to this are obvious: one can look for rhyming patterns or alliterative line beginnings with verse as input to the line-oriented database.

The fifth option is of interest to those concerned with morphological analysis. Here the user is prompted to enter a letter or string of letters as the beginning of a template and then to give a letter or string of letters as the end of a template. What occurs between the first and second string is irrelevant as long as both are contained within the same word. Thus if I were to specify with my Chaucer text that all cases of a word beginning with “y” and ending in “en” were to be found and listed with line numbers then I would obtain a list of all past participles of strong verbs in Chaucer. It is up to the user to formulate the template for a search as unambiguously as possible. Thus in the example just quoted the internal search criteria in the programme are not morphological but purely graphotactical. It so happens that all words starting with “y” and ending in “en” in Middle English are past participles of strong verbs. Were there, however, other word forms which fitted this template then they would also be included in the text file created on searching the line-oriented database.

One of the principal difficulties in automating textual analysis on a computer is the task of disambiguizing one’s material. A means of achieving this is to include the context in which the word one is searching for is to be found. The sixth option in Pat-Mat listed above is intended to do just this. Take such a common word as “that” in Middle English. This can of course be either a relative conjunction or a demonstrative pronoun. However, if, when searching through a database, the word before and after each occurrence of “that” is included in the resulting text file then one can easily recognize what word class form one is dealing with. Note that this option in the programme does not itself decide, on the basis of the context, whether a given occurrence of a specified word belongs to a certain word class or not. This decision must be made by the user. Specifying the context in advance, for example by initializing necessary variables and running a variable-dependent search is not as easy as one might think. Take the example of “that” again. Specifying the word that follows is not enough in English as the two following sentences, which contain the word as members of different word classes, show:

a) He said that good literature is immortal. b) I am aware of that good literature which has been written in Finland.

Suffice it to say at this point that developing quasi-parsing algorithms as criteria in database searches which are anyway accurate is a sheer impossible task. Or one must be prepared to accept a high rate of misses which would, however, mean that manually sorting out different word classes after a search would in the final analysis not represent more work.

1.3. CORPUS

Well what else can one do with LEXA apart from statistical work? There remains the area of indirect parsing, what I choose to call here parsing by assignment, which is quite a promising way for automating lexicological work with texts. For
this one must make use of the third part of LEXA, namely Corpus.

The section Corpus is a database which is intended to be used for storing information about the various lexical items which occur in an input text. To begin with a Corpus database simply appends a word list to itself, placing the words in the first field of the database. The remaining fields of the database are intended to hold specific items of information on lexical forms (see structure of Corpus database on hand-out). Each record in a database then contains a particular word along with possible variants, a full grammatical classification of the word, notes on its meaning, etc. A field for identification number is included in the supplied Corpus database which can be used in later applications or for some sorting purpose or other. Two fields are included for “context” and “attestation” to show where the given word comes from.

The suggested procedure for using a Corpus database is as follows. With Pat-Mat one can determine what the most frequent forms in a text are. These are then analyzed (manually of course) in Corpus and the records containing them are filled in as completely as possible. One of the facilities of Kwic-Kwoc allows one to take a database and filter all multiple occurrences of words in a word list so that with a text like the Chaucer one I mentioned above one can obtain a database which consists of every word which occurs at least once in the text but only listed once. Note however that this does not apply to a lemmatized text but to all possible tokens which may occur in a text. For example, the “unique sorting” option in Kwic-Kwoc will append all instances of a strong verb class to the unique database and not join them to a single lemma before the append procedure is carried out.

The type of database which one creates with Corpus is a working dictionary. The idea behind it is that the user starts building a dictionary of lexical information and each time he analyzes a new text the amount of new classification he has to do decreases.

Consider how this works in practice. One takes an input text, makes a unique database of the words which occur in it and then creates a Corpus database from the unique records database. After this one starts the task of classifying the words which have been adopted into the Corpus database. Once this manual work is done one can then use the working dictionary with another text. Corpus provides the option of merging a word list from an input text with an already present Corpus database, i.e. with one’s working dictionary. This means that one just has to fill in the vacant fields of those records, present in the joined database, which contain words not already included in the working dictionary. By this means the working dictionary becomes more and more comprehensive and the parsing of later texts becomes easier as information on individual words is already included in the dictionary.

In order to maintain a maximum of flexibility I have provided the user with the possibility of editing the structure of databases in Corpus. Say for argument’s sake you find that you need extra fields not included in the Corpus database supplied with the package or you wish to label them differently or change the type of data which a certain field holds. Then you can create a so-called structure database from the supplied Corpus database and edit this. Editing a structure database means that you can add or delete fields, change fields length, change data type and of course change the name of the database. It goes without saying that after creating a structure database from the supplied Corpus database and creating a new customized database from this that the fields names and size are displayed
accurately in the screen masks offered within *Corpus*. The only restrictions are
that fields length cannot be greater than 60 spaces and each can contain no more
than 19 fields. This means that with a lexicological working dictionary which
contains a keyword, its context and its attestation the user can still specify up to 16
fields to contain additional information on a specific item. This should be
sufficient for all practical purposes.

Whatever way one arranges one’s database in *Corpus* one can use that
structure to make a shell database for some other purpose. It is also possible to
specify a database for use, make a working copy for a particular session and only
declare the working copy the archive copy on concluding the work session.

Append facilities allow one to add records from another database to the
current one. Replacing and sorting facilities allow one to replace the contents of a
specified field globally in a database and to sort a database on a certain field with
certain contents. This latter option means that one can create excerpt databases. To
give you a concrete example. Say after working with the Chaucer text mentioned
already one wanted to extract from one’s working lexicological dictionary all the
records which had verbs as their keywords. One simply excerpts the current
database on the basis of the field “Word Class” having the contents “Verb”; the
field “Sub Class” could further be used to extract all cases of strong verbs from
this database, etc.

Another important function within *Corpus* is the possibility of reindexing
one’s database. As supplied, the *Corpus* database is indexed on the keyword field
which means that the records appear in an alphabetically sorted form according to
the contents of the keyword field. However, one can choose any other field as the
keyfield when reindexing the database so that the database then appears in a
different form to the user. Reindexing is of course also possible with user-created
databases.

When using any database standard functions are made available to the user,
such as jumping to a particular record, moving to the beginning or end of the
database, editing a record, appending a blank or number of blanks, tagging or
untagging records for deletion, displaying the records of the database in either a
one-record-per-screen mode (with all fields) or in a one-record-per-line mode
(for a selection of fields).

2. Conclusion

Finally I should mention some additional features of all the three subparts of
*LEXA*. There is a DOS interface supplied with each. This means that the user can
temporarily leave *LEXA* and go to the operating system. He can also run various
DOS commands directly such as restoring or backing up files, directory sorting,
etc. One can furthermore run one’s own word processor from within *LEXA*. This
means that if one wishes to examine the result of a search or sorting operation
(inasmuch as this has created a text file) one can do so without leaving *LEXA*.

Printing facilities are also offered. These are perhaps not as
comprehensive as with other database programmes (such as those additionally
included in *A Computer Package for Linguists*). The main outlet for data
produced within the *LEXA* environment is provided by text files. Nonetheless one
can echo search operations to the printer. One can print Kwic and Kwoc indices of
various kinds, statistical information, etc.
So-called gateway facilities are also provided for. By this is meant that one can export database information to a text processing system. Thus it is possible to convert one’s entire working dictionary in Corpus to a text file and process it further with a normal text editor.

All the databases (except the structure databases which the user may create himself) contain a field for comment (a so-called memoedit function). This allows one to tag a stretch of text on to each record of a database. This text is freely definable by the user and can be of any length. Moving around the memoedit field is done with the cursor keys like in any other text editor. The window which is provided for the memoedit function scrolls vertically so that the user can view as much text as he wishes by simply using arrow up and arrow down.

Let me stress in conclusion that my aim with LEXA has been to provide as open and user-friendly a system as possible. No knowledge of programming is required and only the basics of database management must be understood. Whatever one puts into LEXA can be retrieved again (you can write your databases back to text form); all the results of the various operations within LEXA are available in text form and can be processed without further alteration by any word processor. Maintaining a maximum of generality has meant that I have had to reject include too specialized features. The user who finds that certain of his needs are not satisfied by what is supplied can attempt to create the necessary facilities himself within the LEXA environment (for example by creating and indexing his own database within Corpus) and apart from that he can always continue processing his material afterwards in a different environment and transfer into the latter the results of those operations which he has carried out within LEXA.