# The linguist and the micro-computer : the encoding of a medieval corpus

by

Judy PORTIER
Pieter VAN REENEN
*AMSTERDAM*

1.- Why use a microcomputer

Many linguists interested in computational linguistics leave the programming to professional programmers. Sometimes, they are even advised, in no uncertain terms, to keep their hands off the computer and the programming. This advice will be followed the more willingly, if the linguist is put off by the technique of programming and by the machines involved. However, this does not always result in an optimal use of the possibilities of the computer. We do not think that the linguist has to write all programs himself, but we are convinced that the linguist who is familiar with the technique of programming can make better use of the computer facilities. He learns what kind of problems he may solve by means of the computer and how to formulate these problems, i.e. he learns to think in terms of unambiguous algorithms. The linguist will be stimulated to formulate his hypotheses in an explicit and exact manner, for vagueness and ambiguities will result in output from the computer he did not intend to ask for. The microcomputer is an excellent tool by means of which the linguist can not only learn the technique of programming but he will also be able to use the programs on his research projects.

The microcomputer may serve to process data and to test complete theories, isolated hypotheses, or even hunches, so it can be used for extensive and for small-scale research projects. The microcomputer is a cheap alternative for the minicomputer or a larger system : for about $ 6000, - a complete configuration including disk unit and printer can be bought. Once this amount is invested one has at his disposal an unlimited amount of CPU-time, for which one does not have to pay. When the linguist works with a mini-computer or a larger system, he (or his research unit) will have to pay for the CPU-time used, since it is not his or (its) own machine. This will often amount to a considerable charge to the research budget. If there is no money left, there can be no more research.
Furthermore the user of a microcomputer has direct contact with the data by means of his video screen (he works on-line). He can control his output and interrupt the running of a program the moment he observes that something is wrong. Users of larger systems have to pay extra for working on-line. Since money for research is becoming scarce, and budgets of research units (of Faculties of Arts) tend to decrease, investment in research projects will have to be as economical as possible.
We therefore advocate the introduction of the microcomputer in linguistic research units (in Faculties of Arts and the like).

2.- The AVT project

The project started in august 1980 and is a joined project of the Instituut voor Nederlandse Lexicologie (INL) (Institute for Dutch Lexicology) in Leyden and the Department of General Linguistics of the Vrije Universiteit (VU) (Free University) in Amsterdam. Participants, besides the two authors of this paper, are W. Pijnenburg and P. van Sterkenburg.

The project has both a long term and a short term purpose :

- the creation of a linguistic data-base of Early Middle Dutch; (long term). Our intention is to

make the material accessible to phonological, morphological, syntactical and lexical analysis;

the production of an Atlas of Early Middle Dutch Language Variants : Atlas van Vroegmiddelnederlandse Taalvarianten (AVT) (short term : scheduled for publication at the end of 1982). As is usual in medieval texts, tokens exhibit abundant graphic variation. As far as phonological variation is concerned, we accept the hypothesis that the spelling of the documents reproduces important aspects of the pronunciation. We therefore consider these documents to be appropriate material for dialectical and diachronical phonological analysis.

The corpus we work on consists of more than 2000 official documents (charters) and a glossary, written between about 1230 and about 1325. Besides the glossary, the corpus consists of inventories, rolls, bills, testaments and customaries. The majority of documents is known as the "Corpus Gysseling", published in Gijsseling (1977).

We chose this corpus because :

- the language used in the documents is located in time and space : generally we know when and where (and even by whom) these documents - which are originals - were written;
- the documents cover an extensive part of the geographic area in which the Middle Dutch language (or dialects) were spoken. Figure 1 shows the geographical distribution of the documents.

The number of word forms (occurrences) is about 1,000,000; the number of differently spelled word forms is about 50,000. For budgetary reasons we decided to realize our project on a microcomputer. The configuration we work with consists of a Exidy Sorcerer microcomputer, Z80 CPU, 64 Kbyte memory, 2 diskdrives for floppy disks 5 1/4", with a total capacity of 630 Kbyte formatted on-line storage. Our programs are written in Pascal (Pascal MT/3.2) and in Z80 Assembly language and work under the CP/M operating system.

## 3.- Activities

The operations we have to carry out to realize our project are partly determined by the fact that the data was already on magnetic tapes before we started, but not specifically processed for linguistic research. When data entry is carried out specifically for linguistic purposes, provisions can be made to facilitate this research : words can be marked by unique boundaries, (some) linguistic codes can be added, etc. However, manual input of data is time-consuming and labour-intensive, and therefore too costly for many research budgets.

Fortunately more and more printing-offices are making use of computer controlled printing procedures, so lots of recently published data will be available in a computer readable form at low cost. Not recently published text can be made available by an optical reading machine, which recognizes printed characters and stores them on a (magnetic) recording medium.

So, although our situation may be rather unique at the moment we think that our experience can be useful to others since we are convinced that in future more linguists will make use of material originally processed for other purposes.

In the remainder of this paper we describe the strategy we use for encoding the corpus and the solution we found for a fast and efficient retrieval of information.

### 3.1.- Encoding strategies

For purposes of linguistic research the words must be given a judiciously sophisticated code to make it possible to get information about the phonological, morphological, syntactical and lexical phenomena in our corpus.

It is of course possible to add the code to the lexical items by hand, but we rejected this solution for the following reasons :

- it is time consuming; it would take four people at least a year (we had 1,000,000 word occurrences) and as none of us can work fulltime on this project it would have taken much longer.
- we would not have learned anything of use for a similar job in the future; having finished this monkish work, we would still need the same amount of effort and time to encode, say, a corpus of medieval literary texts.

It is also possible to let the computer add a code to the most frequent word forms and encode the rest by hand. Then 366,403 occurrences (64 different word forms) will be encoded, i.e. ca. 37% of the total. Which leaves 63 % occurrence (still about 50,000 different word forms) to be done. We do not think it possible to encode all occurrences (100%) automatically, but we ought to be able to handle more word forms (perhaps 70-80%) on the basis of linguistic analysis. 50% of the word forms are hapaxes (i.e. they occur only once in the corpus). We therefore have to try and formulate rules by means of which word forms can be combined.

#### 3.1.1.- Application of rules

Middle Dutch texts can show variation in spelling, the linking of lexical items, clisis, assimilation, syncope, etc. If we succeed in formulating rules for these phenomena, we will not only be able to combine the different forms belonging to the same lemma and so save time, but the rules may also provide us with ideas for later, when we want to look at distributional characteristics.

The rules we formulated must be divided in :

- context free rules, which can be applied independent of context;
- contextual rules, application which is dependent of context.

Because we need the original spelling of the word to encode the original corpus, we created a new file in which each word form was represented both by the original form and by a dummy (or a copy of the original form) to which the simplification rules could be applied.

To check whether application of a formulated rule in fact produced the expected result, the result must be listed. If an error is discovered one has to decide between the following alternatives :

- the rule should not be applied or should be reformulated; because errors are systematic errors they can easily be corrected;
- the rule can be applied, but a manual correction will have to be made;
  since our main purpose is encoding and not the formulation of rules, we prefer efficiency over elegance and accept manual correction when the error rate is low.

When applying rules, the order in which the rules are to be used must to be taken into account.

We will give examples of formulated rules and their application on some word forms representing the word "iar" (year) in Middle Dutch.

Context free rules :

1. { j/y → i
   e.g. wherever the letters "y" or "j" are used the letter "i" can be used. Therefore, the letters "y" and "j" in the wordforms can be replaced by "i";
2. h → [0]
   in many texts, use of the "h" is arbitrary. Forms like "ihar" appear besides "iar" (year), so one can decide to delete the "h" altogether.

Contextual rules :

3. { e/i → [0] iff V/– –
   long vowels can be represented by the vowel (V) followed by "e" or "i"; so the letters "e" and "i" can be deleted if and only if they are preceded by a vowel.
4. c → ts iff #/ – – /h
   "c" has to be replaced by "ts" if and only if it is preceded by a word boundary and followed by "h"

Of course, rule number 2 can not be applied before application of rule number 4.

3.1.2.- Defining lexical items

After the word forms are simplified a listing can be produced containing the different dummy word forms. The next step is to formulate definitions by means of which the lexical items can be identified. A reverse listing of the above mentioned dummy word forms can be very helpful, because certain

suffixes can indicate the class of the word, e.g. the suffix "-scap" is always part of a noun.

ARTICLE = {d/ds/ts} iff #/ – –/NOUN
"d" or "ds" or "ts" is an article if and only if they are preceded by a word boundary and followed by a noun.

NOUN = {iar/scap} iff – –/{e/en/es/s/[0]}{#}
a dummy word form containing "iar" or "scap" can be identified as a noun if and only if it is followed by "e", "en", "es", "s" or nothing, which is followed by a word boundary.

FLEX.NOUN = {e/en/es/s} iff NOUN/– –/#
"e", "en", "es" or "s" are flexion morphemes of a noun if and only if they are preceded by a noun and followed by a word boundary.

The computer will now be able to identify the lexical items and we can instruct the computer to split article and noun if they are tied in one wordform. At a later stage this word splitting will have to be executed in the original file. We must therefore prepare to let the computer know this in due time. We will insert the "+" in the original file later to supply the information that the words were tied together originally. When the lexical items are provided with a code, we can instruct the computer to add a code to every dummy in the file. This amounts in fact to having coded all corresponding original forms.

For the encoding one can choose between the following alternatives :

- Only those forms must get a code which can be classified unambiguously as a member of just one word class.
- Forms which can be a classified as a member of more than one word class can also get a code if the form occurs most frequently as a member of one word class.
  e.g. "Ende" can not be unambiguously classified. It can belong to the wordclass 'conjunctions' (in which case it would be translated as "and"), but it can also belong to the word class 'nouns' (when it would be translated as "end"). It is, however, reasonable to expect that the frequency of its occurrence as a conjunction will be much higher. One can therefore decide to encode all occurrences of "ende" as a conjunction and correct the errors semi-automatically : if "ende" in the original file is preceded by an article then it has to be encoded as a noun.
  CODE-X –> CODE-NOUN iff {CODE-ARTICLE} {ende} / – – – –

The unclassified (uncoded) forms can be deleted from the wordfile. They can be stored in a file containing the ambiguous forms.

3.1.3.- Inserting the code

When we finally have at our disposal a file consisting of wordforms + codes, a new file containing

only alphabetically sorted original forms + codes can be created. The computer can then quickly find out if a word read from the original file is represented in the alphabetical list. If so the code can be attached to it. If we want, we can instruct the computer to display the word with its context in case the word is not found in the list and to wait until the code is typed at the keyboard. Another solution is to create a file containing the homographs with the codes possible for the specific homographs. When these codes are displayed as well when a homograph is found, input can be facilitated.

3.2.- Retrieving formation

When the corpus will be encoded we want to be able to retrieve information from our data base. For the planned atlas it will not only be necessary to get information about the linguistic units themselves, but we will also want to know where and when a specific wordform was used. We want to ask questions like :

- What types with linguistic property x are used ?
- What types are used in linguistic region x ?
- In which linguistic regions type x is used ?
- In which scriptoria type x is used ?
- How many times are the types x and y used ?

The documents, which were already dated, were numbered as well, so that they can be uniquely identified. For our questions concerning the location we divided the geographical area in linguistic regions. Figure 1 shows the regions we distinguish.

As the file is very large, it takes quite a long time to get the information asked for if the computer has to read every item in the data base, even when a specific document could be skipped. We found the following solution to minimize the time needed to find information. The file containing all occurrences + codes (file0) needs to be preserved for further investigation, but as it is not necessary for the questions we want to ask to maintain each occurrence of the word forms, we created a new file (file1) in which for each document the following information is stored :

- a number for identification of the document;
- an alphabetical list containing the specific word forms occurring in that document;
- for each different word form : - the code belonging to that word form,
  - the number of occurrences in the original document.

Thus the number of occurrences can be diminished and consequently the time needed to retrieve information about the occurrence of wordforms. The alphabetical sorting facilitates the search in the file for a specified type, so time is reduced even more.

A further reduction of time can be achieved by using the facilities of our disk operating system. The operating system makes it possible for the disk head to be moved directly to a specified location on the disk. So if the location of the information wanted is known, it will not be necessary to read the entire disk to extract that information. If we know from which document we need the information and where that document is stored on disk, we can minimize the time necessary for retrieving that information.

We therefore created another file (file2) containing :

- the identification numbers used for the documents;
- for each document specifications like :
  - the identification number of the diskette on which the document (file1) is stored,
  - the first and the last address on the diskette used for the document,
  - the year the document was written,
  - an identification number for the centre at which the document was written,
  - an identification number for the linguistic region in which the centre is located (see figure 1).

Selection of the documents to be considered can be carried out very fast by ordering the computer to look in file2 and select those documents that satisfy certain specified qualifications. The list of documents is sorted according to disk number and the address of the first location on the disk used for the document. After a diskette which was indicated by the computer, is inserted in the disk drive, the first location number can be passed to the operating system. Now the computer can start reading data in the document without spending time reading data that can be skipped.

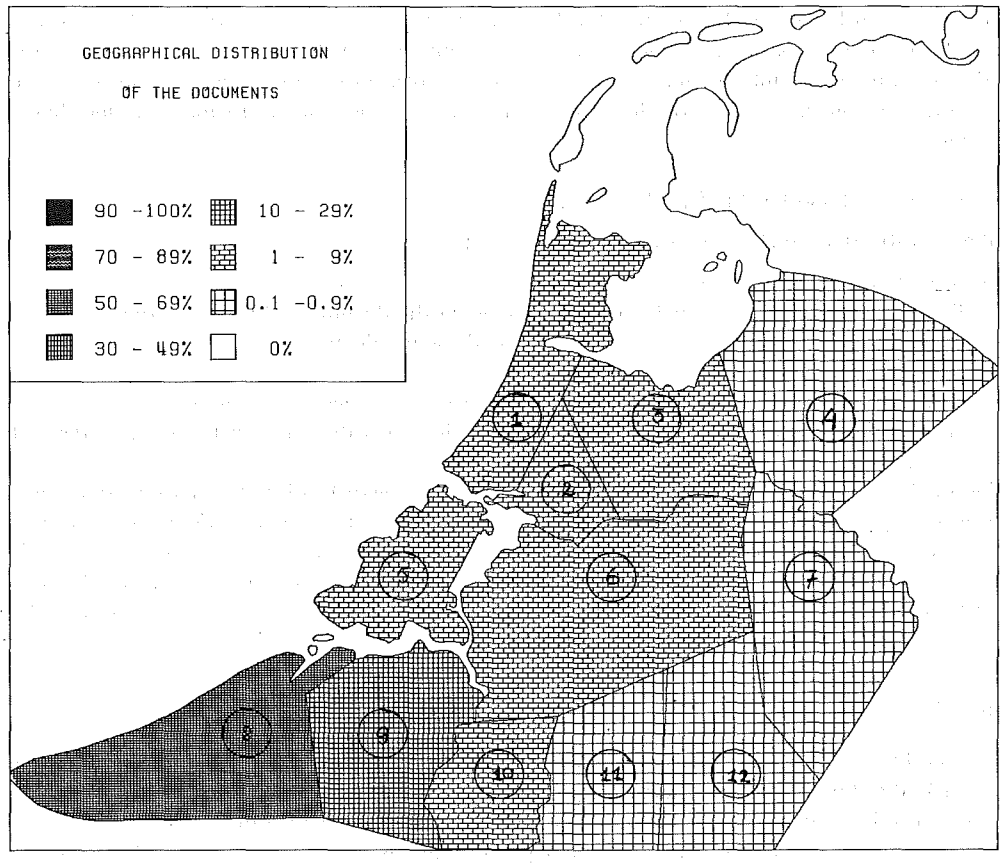
## 4.- Conclusion

After we will have realized our project we will not only have gained in experience with regard to the computer facilities but we will also have enlarged and made explicit our knowledge of Middle Dutch. Besides the encoded corpus, we have at our disposal a list of 50,000 wordforms + codes, and a list of linguistic rules and definitions, which can be applied to Middle Dutch word forms. These lists can not only be used for a similar project in the future, but they can also be useful when we want to write a Middle Dutch grammar.

## NOTES

GYSSELING (1977), MAURITS, Corpus van Middelnederlandse teksten (tot en met het jaar 1300) uitgeg, door --, m.m.v. er van woordindices voorzien door Willy Pijnenburg, 's Gravenhage, Martinus Nijhoff, 1977.

Figure 1




GEOGRAPHICAL DISTRIBUTION OF THE DOCUMENTS

| | | | |
|---|---|---|---|
| 1. HOLLAND-WEST | 6.9% | 7. NEDERRIJN | 0.3% |
| 2. HOLLAND-OOST | 2.9% | 8. VLAANDEREN-WEST | 56.1% |
| 3. UTRECHT | 1.4% | 9. VLAANDEREN-OOST | 19.9% |
| 4. OOST-NEDERLAND | 0.2% | 10. BRABANT-WEST | 8.2% |
| 5. ZEELAND | 2.1% | 11. BRABANT-OOST | 0.3% |
| 6. BRABANT-NOORD | 1.3% | 12. LIMBURG | 0.4% |

| dummy | dummy | code | code | original form | changed original |
|---|---|---|---|---|---|
| | iar | | | iar | |
| | iar | | | iaer | |
| | iar | | | ihaer | |
| | iar | | | ihar | |
| | iar | | | jar | |
| | iar | | | jaer | |
| | iar | | | jair | |
| | iar | | | jhaer | |
| | iar | | | yar | |
| | iar | | | yaer | |
| d | iar | | | diaer | d+ +iaer |
| d | iar | | | diar | d+ +iar |
| d | iar | | | djaer | d+ +jaer |
| | iare | | | iaere | |
| | iare | | | iare | |
| | iare | | | ihaere | |
| | iare | | | jaere | |
| s | iars | | | siaers | s+ +iaers |
| s | iars | | | sjaers | s+ +jaers |
| ts | iars | | | chiaers | ch+ +iaers |
| ts | iars | | | chiars | ch+ +iars |

Figure 2: An example of the simplification of word forms denoting 'year'

```
-------------------------------------------------------
document number ! specifications
-------------------------------------------------------

word form          code       frequency
word form          code       frequency
word form          code       frequency

-------------------------------------------------------
document number ! specifications
-------------------------------------------------------

word form          code       frequency
word form          code       frequency
word form          code       frequency
word form          code       frequency

-------------------------------------------------------
```

Figure 3: example of the structure of file1

| doc.nr | disk nr | disk addr | year | region | scriptorium |
|---|---|---|---|---|---|
| 3 | 1 | 35 | 1236 | 8 | 9 |
| 4 | 1 | 50 | 1237 | 8 | 7 |
| 5 | 1 | 70 | 1250 | 5 | 6 |
| 6 | 1 | 105 | 1250 | 2 | 3 |
| 7 | 2 | 1 | 1250 | 3 | 4 |

Figure 4: example of the structure of file2