Transforming machine readable sources

Werner, Thomas

Historical Social Research, Vol. 16 — 1991 — No. 4, 62-73

Transforming Machine Readable Sources

Thomas Werner*

Abstract: The advent of the big statistical packages has in recent years lead to a de-facto standardization of numerical data: as all the large-scale statistical systems accept roughly the same input conventions, the exchange of date between them has become increasingly easy. Text based, data base related or image oriented projects in history do not, so far, share that benefit: her the exchange of data between individual projects and the utilisation of data prepared by other projects is still a major problem. This paper considers a system for easy transformations of formats between various software environments. While the following considerations are independent of the currently much discussed Text Encoding Initiative (TEI) the might be seen as the implementation of a tool, which could handle the standards which are proposed there, eases, however, also the transformation between text base and data environments.

Introduction and Background

A transforming program applicable to the logical format of machine readable sources as they are used in the historical sciences as well as in the humanities in general has to offer solutions for two problems:

1. first, it must provide suitable techniques to convert data from one software environment to another;
2. secondly, with regard to the contents, it must take into account the peculiarities of historical sources.

The aim of this contribution is to discuss the possibilities of a combined implementation of these two tasks and to describe a system called StanFEP - Standard Format Exchange Program.

* Address all communications to Thomas Werner, Max-Planck-Institut fur Geschichte, Hermann-Fôge-Weg 11, 3400 Göttingen.

StanFEP was developed at the *Max-Planck-Instituí für Geschichte* in Göttingen (1) in the framework of a European research network - the historical Workstation Project« - aiming, among other things, at the provision of existing data bases for circulation among interested scholars. In view of its origin, StanFEP is oriented towards the demands of historical requirements; it is closely related to a data base management system (DBMS) known as Κλειω. Nevertheless, StanFEP is a stand-alone system, written in »C«, and thus hardware independent.

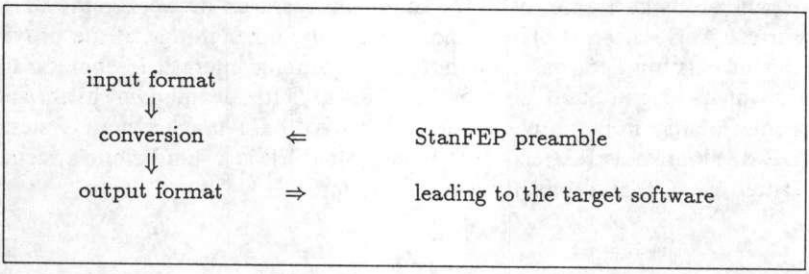
Converting Data

Assume you want to study a medieval German text, say a collection of sermons, using the facilities of electronic data processing. The following might occur: the source already exists in machine readable form, and let us suppose that the responsible scholar would place the data at your disposal. A pleasant situation, indeed, but nevertheless generally associated with great problems. The machine readable text of the sermons is, namely, prepared for the requirements of a specific DBMS, which does not - for various reasons - meet your needs. What would, then, have to be done to make the data readable for another favoured system?

- The coding of, for example, diacritics would have to be changed according to a formula defined by you or according to proposals as they are prepared in the form of SGML.
- Control sequences indicating a new strophe or verse would have to be adapted to the form needed by the target system.
- Possibly portions of the text or other properties of the data would have to change their position. This could lead to a completely new arrangement and organization of the material.

In most cases, a conversion of *the format* - that is, as we understand it, the characteristic design, the principles of organizing the raw data and the control sequences required by a particular DBMS - would be necessary.

StanFEP is software especially developed for such purposes. It makes possible the transformation of raw data (2) from any arbitrary input format to any selected output format. Both the input and the output format have to be expressed in terms of a specially designed meta-language. The implementation of this meta-language in StanFEP guarantees a »neutral« description of many kinds of raw data - i.e. a description that does not apply to only one or two software systems, but which allows a great variety of expression. The description of the material takes place in so-called *preambles.* The StanFEP preamble is, so to speak, the connecting link between the formats of the input on one hand, and the ouput data on the other.



A sketch of the transformation process would look like this: The basis of the conversion is formed by two components; one file containing the raw data, and another containing the preamble including a reference command to the input data. Furthermore, the preamble contains all the instructions written by the user in StanFEP's language and describing, in general, two points:

- What does the input format look like?
- What is the output format intended to look like?

When StanFEP reads the preamble, the system becomes active, gains access to the raw data, transforms this material, and writes the result in a third, user-defined file.

The facility to describe data formats in terms of a meta-language is a useful instrument, especially in regard of one aspect: the range of possible target systems is wide. So, there is the opportunity to transfer *one* given machine readable source not only to *one* well defined target system, but to a number of different systems. This concept of StanFEP offers, for example, the possibility of generating one output file leading to a word processing system, another leading to a DBMS, a third leading to a concordance program, and so on. For these purposes it would be sufficient to formulate a number of preambles, each containing exactly the same description of the input format and, secondly, a specific description of the output format.

The various kinds of formats we are handling can be divided into three groups. Approximately, every DBMS requires the raw data of one of the following different types:

1. *Fixed format:* the structure of the data base is stipulated by a convention of a fixed field-length and, furthermore, a strict order of occurrence of the fields. No specific control sequences are visible within the text.
2. *Structured format:* the data is composed of fields of variable length,

identified and separated from each other by explicit symbols. The order of occurrence of these fields may be predefined or may be indicated by a special name, a so-called label. Often in this kind of data a number of fields constitutes greater entities, groups, which again need signal signs as identification.

3. *Preedited format:* the data material consists of a pure text, where some portions are marked by inserted symbol signs. The design of this format is neither restricted by a firm field-length, nor by an order of occurrence. The input text retains its character as a running full text; it is not structured in a conventional sense, but, in a similar way, by the fact that some marked portions of the text have a special status in comparison with the rest.

StanFEP is able to »bridge« two different types of format as well as different variants of the same type, say the conversion of one fixed format into another fixed format. Regarding the way this is achieved one fundamental remark should be made: It is not possible to give StanFEP global instructions, such as: »Produce a structured format out of the format in question«, or: »Create a representation of the raw data that is the one expected by the target software system X or Y.« Rather, the significant signs and parts of the text in both the input and the output format have to be translated into a language which is understandable for StanFEP. Let us now have a closer look at this language and the main features of the system.

1. *Character changing functions:* At the level of characters, StanFEP supports trivial changing operations. The system accepts instructions relating to a single character as well as to strings of characters, where the length of the strings is, in principle, unlimited. This feature is comparable with the changing functions of a comfortable text editor. One advantage is that StanFEP processes an unlimited number of operations in the course of one and the same transformation. Furthermore, the command language is easy (the basic form of this instruction is: `"string" = "string").`

2. *Strings and Patterns:* On the level of strings StanFEP is able to recognize *patterns.* The ability to match this kind of character strings and to take them as a basis for further transformations, processed according to user-defined rules, is an important feature of StanFEP. These operations are controlled by a module implemented in the system that activates a »subsystem« called CMATCH. CMATCH is an instrument for managing higher pattern matching functions. Inspired by a comparable system, SNOBOL4, CMATCH is written in standard C. (3) Unlike SNOBOL4, this system utilizes a control language that makes knowledge of the internal functions of the program unnecessary. For

the user this conception has the decisive advantage that CMATCH can be applied to describe patterns in a comfortable and easy way. Patterns are expressed in terms of the StanFEP syntax and not in a language only understandable for a programmer. Thus we expect the implementation of CMATCH as a powerful instrument of pattern matching in combination with user-friendly applicability to lead to a significant increase in feasible transformations. Some examples may support this assurance.

With the help of pattern matching it is possible to split a running text at the occurrences of punctuation marks. An instruction like the following is easy to carry out: »Whenever in the data there is a point or a question mark or an exclamation mark followed by a blank and a capital letter, then split the text and assign the sentence to a separated field within the data base.« The necessary description for the matching of this pattern looks like this:

```
#pattern=textbreak,
#definein="signset1' 'signset2";
#signset=signset1, #definein=".?!'";
#signset=signset2, #definein="ABCDEFGHIJKLMNOPQRSTUVWXYZ'";
```

In this case pattern matching is just used to search for strings within the text. Found entries are taken as a basis for intended transformations but the strings themselves remain unmodified.

More sophisticated are those pattern matching functions supporting string manipulations. An example is the following double representation of an antique inscription. It was the intention of the author, in preparing this data set, to reconstruct approximately the state of the preserved fragments. The starting-point of the transforming process is the full text which is equivalent to the printed edition:

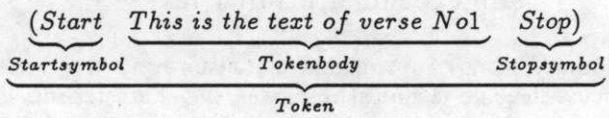
Book$Ti(berio) Claudio Drusi f(ilio) Caiesar^Augusto Germanieco\ pontifici maximo trib(unicia) potesetat(e) XI\ co(n)s(uli) V imp(eratori) XX ... patri paetria^ senatus populusquee Romanus qeuoa] reges Britanniaie XI devictos sinee\ ulla iactura in deditionem acceperite\ gentesque barbaras trans Oceanmume\primus in dicionem\ populi Romani redegerite

```
Stone$
$       Ti Clau_____________sari
$       Augu_________co
$       pontific_________________tat XI
$       cos V im_________________triai
$5      senatus po_______ Ro______uod
$       reges Brit_____ XI __________
$       ulla iactur_____________________
$       gentesque b_________________
$       primus in dici_____________________
```

What happens? First of all, StanFEP splits the text along the linebreak symbols, the vertical dashes. More surprising is the new layout of the text. Here, all those portions of the running text between round brackets are lost. All portions between square brackets have been processed according to the instruction: »Replace each character which occurs after an open bracket and before a closed one by exactly one underlines The command seems easy, but in fact, we have here a typical situation calling for pattern matching tools.

3. *Text and Data Structures:* Converting data, especially into structured formats, often means that entire logical units must be moved within the text. Sometimes they exchange their position with other units, sometimes they have to be extracted from their original surroundings and be put at another point of the data structure. The automatic repositioning of text entities we call *relocation.* StanFEP supports this kind of transformation in several ways.

The pole of every relocation instruction is a single text unit - we call it a *token.* Such a token could be a verse of a poem, the date of an event, or the entire text of a medieval charter. Each token is constructed according to the following scheme:



StanFEP reads *every* input format as a sequence or a nested system of combined tokens: Tokens may have subtokens, which have subtokens themselves, and so on. The logic behind the transform process is simple: The underlying principle of the input transformations holds good with the

obvious differences during the connected operation, the production of the transformed text. StanFEP operates on the basis of a single token and treats them separately as the fundamental bricks of the text. The target text, then, is created by putting one brick after another, or by putting them in exactly that position where the user wants them. The user has to a high degree the chance to affect the result of any transformation, that is the construction of the intended target representation. Formulating the StanFEP preamble means setting up a system of rules, which includes the options to perhaps ignore a specific token, or on the contrary to replicate a token in order to treat it differently. Possible instructions for a complex relocation might be:

- relocate one token after another token;
- relocate one token after a whole chain of tokens;
- relocate one token before the first occurrence of a defined token. (4)

Especially in the historical sciences, where our basic material, the sources, is of extreme variability, the possibility of automatic relocation is a permanent necessity. The structures of given texts can be modified and, sometimes, reorganized according to the requirements of the selected target DBMS, as well as to the intentions of the user.

Summarizing the first part of this contribution it should be noted that StanFEP provides efficient tools for three main aspects:

1. first of all, at the level of characters, changing functions;
2. secondly, at the level of logical units, the various operations on strings and patterns;
3. and, thirdly, at the level of text and data structures, the possibility to relocate defined units within the text.

StanFEP is, as we think, superior to other existing software packages, because it provides an overall solution for the points mentioned. The system offers wide ranging possibilities in the field of data conversion.

Fulltext and Structured Text

In my preliminary remarks I said that a transforming program should support, besides adequate technical resources, the characteristic working techniques of electronic data processing in the domain of the humanities. One of these typical methods and the resulting demand for an adequate software solution will be discussed in the remainder of this paper.

DBMS, in general, only work as well as their raw data are organized. Of course, there is no basic principle prescribing the design of machine readable sources, but there is one common and recurrent necessity: in many

cases, the double representation of a given source, first as a fulltext and secondly as a structured one seems advantageous for later processing by the selected DBMS.

Two examples may help illustrate this assertion:

- A historian studying the social behaviour of a community could be interested in the way how persons acted in a pressure situation, say a trial before a church court. He would, on the one hand, read the court records with respect to strategies of argument, the way judge and defendant communicated, and so on. On the other hand, he would examine the social status of the participants, indexing their names and collecting information about their occupation and career. This kind of research would make fulltext retrieval necessary, as well as a parallel evaluation of the separate information items.
- For a scholar, working in the field of linguistics, it could be advantageous to administer both a running text and a structured text within the same document of a data base in order to check, for instance, first the use of deictic words in their context and afterwards to evaluate the percentage of parts of speech used.

The common pattern of these and comparable techniques is that the user of a DBMS searches for information in the fulltext with more or less complex criteria. Entries found are then evaluated, in isolation or in comparison with each other. That is why they have to be administered separately in fields or other information units.

When we argue that the double representation of a source is often a desirable state of affairs, it must be said that hitherto in most cases there was no reasonable chance of this aim being achieved. An important obstacle is the lack of time and money necessary for data entry that captures the material twice.

These circumstances were considered seriously during StanFEP's conception. As a result there are suitable tools to make the preparation of raw data as optimal as possible and so to lay the foundation for effective processing by the selected DBMS.

In cases where it is intended to create documents containing a multiple representation of identical or partly identical information, we propose the following procedure:

1. *Data entry*: The basis for data entry is the original source, usually in the form of a running text, without any manipulation. The machine readable source should be faithful to the manuscript or edition. How the data are entered into the computer depends on the quality of the written text. If possible, it should be scanned via a data entry machine (such as the KDEM); if not, typing it in the conventional way will be unavoidable. An example:

This is a faithful machine readable copy of a given source. The running text contains some notable information that should be marked. Of importance are, for instance, historical persons, like Kaiser Friedrich II. Barbarossa, who died 800 years ago, on the 10th of June, 1190, in the river Saleph, Turkey.

2. *Preedition:* Now, the historian »preedits« - as we say - the pure text according to his own conceptions. All those passages containing important information are marked by user-defined symbols. This prevents decomposition of the original source and guarantees furthermore a dynamic and flexible approach to the specific characteristics and peculiarities of different sources. It must be noted that all start- and stop-symbols have a different status in comparison with the other signs. They are valid as control sequences and in this way they constitute the design of the *preedited format:*

(doc *This is a faithful machine readable copy of a given source. The running text contains some* (inf *notable information* inf) *that should be marked. Of importance are, for instance, historical persons, like* (p (t *Kaiser* t) (n *Friedrich II. Barbarossa* n) p), *who died 800 years ago, on the* (date (d *10th* d) *of* (m *June* m), (y *1190 y)* date), *in the river Saleph, Turkey,* doc)

3. The *StanFEP preamble:* In a third step StanFEP has to be involved in the transforming process. The format of the preedited text, that is the strings of inserted additional characters and their meaning, has to be described in terms of the StanFEP-syntax. In addition the system receives instructions advising it what has to happen to the source, and thus qualifying the intended target design of the text. Briefly: The user formulates a preamble oriented towards the underlying machine readable source. Some possible instructions might be, for example:

- »Regarding a first target representation of the source, write down the original running full text and forget about the inserted marking symbols« The system would now ignore the strings `(inf`, as well as `inf)`, furthermore `(p`, `(t`, and so on. It would only pay attention to the symbols `(doc` and `doc)`, indicating that the included passage constitutes a unit of the full text.
- »Regarding a second target representation of the source, extract all those portions specially marked by inserted symbols, and relocate them in separate fields before the full text.« For this, the control sequences required by the selected DBMS have to be well defined, by using, for example, the following part of the preamble:

```
#name=information,
#startin="(inf", #stopin="inf)"",
#startout="//information=", #stopout=" "",
#before=fulltext;
```

4. The result of the described procedure would be the intended double representation of the original source, for example:

```
document $
person$title=Kaiser/name=Friedrich II. Barbarossa
date$day=10th /month=June/year=1190
inf$notable information
fulltext$This is a faithful machine readable copy of a gi-
ven source.
The running text contains some notable information that
should be marked. Of importance are, for instance, hi-
storical persons, like Kaiser Friedrich II. Barbarossa,
who died 800 years ago, on the 10th of June, 1190, in the
river Saleph, Turkey.
```

The design of this raw data set would be sufficient for an evaluation in several dimensions. A DBMS suitable for the demands of historical research could take, for example, the fields containing name material as a basis for indexing operations, the fields containing dates for the computing of time intervals, and so on. And it could take the full text representation as a basis for context oriented information retrieval.

The possibilities we have sketched for generating raw data with the help of a transforming process may have shown some worthwhile implications of the handling of machine readable sources. This kind of data preparation is, of course, quite different from the conventional kind. The aim in the form of the raw data file has to be achieved indirectly, making a detour via a computer program. This program must be be learned, so that the use of StanFEP presupposes an input of time. This has to be taken into account, as well as the consideration that the application of StanFEP should not be undertaken for its own sake. StanFEP is a »preprocessors When the intended raw data file exists the work of StanFEP is done. Afterwards, the system would only be activated again in those cases where corrections seem necessary or where the amount of source material has increased and new parts have to be transformed. So - is StanFEP nothing but a detour into a dead end street? We do not think so, because some advantages of using this system are obvious. To sum up the main aspects:

- *with regard to the contents and specific working techniques:* The full text, that is, the original text of the source and the machine readable copy that guarantees maximum faithfulness to the source remain entirely intact. This on its own should already be regarded as an advantage, because it is a mark of humanistic research that recourse to the original is of the highest importance for a variety of questions.
- *flexibility*: The decisions how to transform the data material need not

be taken at the start of entering data. Furthermore, decisions concerning the preedition of texts are never definitive so that they could not longer be withdrawn. The possibility of describing data externally in terms of a meta-language enables the user to adjust an existing preamble to new demands occurring in the course of the work. For example, without great problems a source could be adapted to the requirements of a word processing system, even if it was originally intended to be evaluated by means of a DBMS.

- *circulation of preeditions:* Closely related to the flexibility of StanFEP is a further aspect concerning a more ambitious dimension. The preedition of a machine readable source could be a very good basis for a transfer within the scientific community. A scholar working in the same field of research would profit by receiving this data, provided it is sufficiently documented. He or she could then treat the inserted marking symbols as a proposal, which does not prevent easy modification by formulating his own StanFEP preamble. It is to be hoped, from our point of view, that a transfer of machine readable sources prepared for transformation with StanFEP together with the use of the system itself will contribute to intensified communication and discussion among scholars working with the facilities of electronic data processing in the domain of the humanities.

A View to Further Developments

The features described are, at the present state of development, fully implemented in StanFEP. A first version of the system is due for release in August 1990.

Further enhancements have already been initiated. One of them will be sketched at the end of this paper. Concerning the approaches to a running text, StanFEP hitherto has been able to identify defined strings and patterns. Both can be processed with the help of CMATCH. It is now intended to implement a further module in the software package providing matching of »semantic« patterns. For this purpose, an interface will establish a connection between CMATCH and a function library containing thesauri of semantic networks. (5) Here, the user has to define, e.g., fuzzy-weighted relationships between different terms of a word field. Another possibility will be the creation of a thesaurus describing a number of expressions for a thematic item, e.g., »all terms pertaining to >sovereignty< ».

Later, during the processing by StanFEP there will be allowed queries on patterns, which are not defined by the occurrence of a set of characters, but relate to criteria specified within these external libraries.

Notes

(1) The development is sponsored by the German Research Foundation.

(2) That is data prepared for a specific DBMS, administered in a separate file and loadable into the environment of the system.

(3) Like StanFEP, it is hardware independent and, furthermore, a stand-alone system, being controlled by an object oriented language.

(4) For an example, cf. p. 8

(5) A prototyp of this library already exists *(FuzzNet),* but is not yet in production use.