

**The FACIT Project
LIB FACIT/ 1-1044**



Technical Report No 3

**Error Analysis and Correction
in Retroconversion**

Hans Erik Jensen

October 1996

General preface to the FACIT Reports

The present report is one of five presenting the results of the FACIT project.

FACIT is short for Fast Automated Conversion with Integrated Tools. The project has been supported by the EU under the Libraries section of the Telematics Programme. It started in January 1993 and finished in February 1996.

The project has been concerned with two main questions:

1. To determine the feasibility of converting older card catalogues into modern OPACs using scanning, Optical Character Recognition, and automatic formatting into a bibliographic format (such as UNIMARC).
2. To develop a prototype system capable of handling automatic formatting, and automatic or semiautomatic detection and correction of the errors produced by scanning and OCR.

The first objective has been achieved in the sense that the project has shown that such retroconversion can only be expected to be feasible under certain conditions.

The Achilles' heel of fully automatic procedures in retroconversion is still the speed and quality of OCR. And this depends to a large extent on the state of the source material. The project was based on the assumption that commercially available equipment and OCR programs would be able to handle older typewritten and printed catalogue cards in a satisfactory way, so that the main effort could be aimed at formatting the cards, but very much more time has been spent on problems of OCR than was originally envisaged.

The results concerning OCR are based mainly on the use of commercially available scanners and OCR packages in the lower or middle price range (such as seem attractive to most libraries). The results may have been different if more sophisticated (and more costly) equipment had been used, or even custom built equipment. But in general the conclusion has to be that many older card catalogues are not suitable for this type of methodology because of the state of the source material: Yellowed by age, worn, smudged, with

handwritten additions, sometimes swollen or made uneven by dampness, written with a series of typewriters with varying typefaces and with ribbons that are more or less worn out, copied by stencilling etc.

The conclusion is not that the methodology is not feasible at all, but that its application is limited to fairly "well behaved" catalogues. A library wondering whether to apply scanning and OCR to retroconversion should carry out extensive tests in order to assess the suitability of this.

Formatting the cards after scanning and OCR does not, on the other hand, seem to present serious problems, if the output from OCR have a low level of errors. Based on a thorough formal analysis of the catalogue and the rules used in producing it, it will in most cases be possible to write a series of programs specific to that catalogue to do the job. This is confirmed by other projects.

The main focus of the project was to investigate the possibility of producing one application, able to handle a wide range of card catalogues as found in European libraries, avoiding the necessity of writing the formatting programs from scratch every time. This is done by feeding the application a formal description of the catalogue at hand, using a relatively simple formal language. At the same time the application should provide a set of integrated tools for the range of different procedures that go into retroconversion work. The project has demonstrated that this is in fact feasible.

But the work of formal analysis is quite demanding, both in terms of time and the necessary skills and knowledge. And it will have to be done again with each new catalogue, since no two catalogues are exactly alike. This process is needed in order to produce the necessary formal specifications for the formatting programs, both with a system like the FACIT Prototype and with custom built formatting programs. This is definitely a specialist job.

With automatic conversion lot of the costs go into setting up and testing the system with each new library and each new catalogue. This means that this methodology is not suitable for a small or medium size library to handle alone without expert assistance - from a commercial service or a large library that has already done some work in this area.

An important problem that has not been solved in a satisfactory way in this project, is the need for detecting and correcting errors produced by scanning and OCR. The project has investigated various possible solutions, and it seems worth while to pursue this further. Meanwhile corrections will have to be

done by the human operator with some support from the computer.

The second objective of the project, as stated above, has only been partly reached. A software package has been developed that is able to demonstrate the principles involved in automatic formatting of library catalogues and in customizing the procedures for use in libraries with widely different cataloguing practices as well as catalogues produced over time to different specifications. But the package does not include more advanced facilities for error detection and correction, and it still lacks a series of features that are necessary for use in large scale conversion of catalogues.

Nevertheless the results of the project are promising for further development work, and constitute a solid basis for future work by the partners and the subcontractors of the project as well as others. The aim of the published reports is therefore to make available the information generated by the project, in order to help making realistic judgements about the prospects of using the methodology described in a particular library for the conversion of a particular catalogue, and in order to make the information useful for other research and development projects.

The published reports from the FACIT project consists of the following:

Optical Character Recognition for Retroconversion of Catalogue Cards: Hardware, Software and Character Representation. By Niels Erik Wille. (FACIT Technical Report no 1). Statens Bibliotekstjeneste, Copenhagen. October 1996.

The report summarizes the experiences with scanners and OCR programs. Special treatment is given to the question of character sets and representation of characters, since this is normally of great importance in converting multilingual catalogues.

A Framework for the Analysis of Catalogue Cards. By Niels Erik Wille and Vera Valitutto. (FACIT Technical Report no 2). Statens Bibliotekstjeneste, Copenhagen. Revised version, October 1996.

The report describes the problems involved in analysing a catalogue in order to evaluate the feasibility of converting it by automatic means, as well as the formal language to be used in setting up the FACIT Prototype. This information should also be useful for someone aiming at developing similar tools for retroconversion.

Error Analysis and Correction in Retroconversion. By Hans Erik Jensen (FACIT Technical Report no 3). Statsbiblioteket, Aarhus. October 1996.

The report summarizes the issues involved in automatic or semiautomatic error detection and correction, and outlines plans for further development of the Prototype in order to incorporate more sophisticated handling of OCR errors.

The FACIT Prototype. Manual and Documentation. By SYNERGI (FACIT Technical Report no 4). Statens Bibliotekstjeneste, Copenhagen. October 1996.

The report describes the Prototype in detail and the procedures to use when setting up the demonstration version. The level of information is highly technical. Due to a series of limitations the demonstration Prototype is not suitable for large scale conversion work, but using it with a smaller sample will provide a good grasp of the problems and procedures involved in automatic formatting etc.

Retroconversion of Older Card Catalogues using OCR and Automatic Formatting. Project Overview and Final Report. By Niels Erik Wille (FACIT Technical Report no 5). Statens Bibliotekstjeneste, Copenhagen. October 1996.

This report presents the project as a whole and the main results reached. It includes a summary of the information included in the previous reports.

These reports are available free of charge.

A workable demonstration version of the FACIT Prototype is available. This is a combination of a suite of DOS programs and an interface produced as an application for Microsoft Access. The Prototype will run on a PC with Windows 3.11 or Windows 95 and Microsoft Access 2.0 or later versions. The Demonstration Prototype is available free of charge for use in European libraries.

All correspondence concerning the reports and the Prototype should be sent to:

Niels Erik Wille
Senior lecturer
Dept. of Computer Science, Communication and
Education
Building P4
Roskilde University
P.O.Box 260
DK-4000 Roskilde

or posted by e-mail to: new@snow.ruc.dk (Internet)

Information about the project and copies of the reports and the demo-version of the FACIT Prototype are also available on the World Wide Web at the address: <http://www.komm.ruc.dk/FACIT/>

Contents

1. Introduction	7
2. Error types generated by the process	8
3. Types of OCR-generated errors	10
3.1 Spelling-checkers	12
3.2 Possibilities of correction	15
3.3 Outline of the method	17
4. The reformatting method	18
5. Recommendations	19
6. Literature	21
Appendix 1: Terminology and Basic Concepts	23

1. Introduction

The main steps in realizing the retroconversion of a catalogue are the following:

1. Making some preparations for the scanning process, i.e., counting the number of cards pr. batch for control or making other steps to control the process.
2. Scanning the cards batchwise, resulting for example in a TIFF-file.
3. Using the TIFF-file as input to an OCR-package, resulting in a character-recognized file probably with errors.
4. Correction of the errors in the file, resulting in a file ready for automatic reformatting.
5. Reformatting of the file in fields.
6. Check for example, that all cards are read, all mandatory fields are there.
7. Final conversion if necessary from the internal format and character-set used to the format and character-set of the target system, normally a MARC-format.
8. Check that all cards have been loaded up on the system.

Before this production process can be executed, it is necessary to make an analysis to confirm, that it at all is possible to use this technique to convert the material to machine-readable records. The outline of this necessary analysis has been described elsewhere, so here we will describe the critical steps assessing, how many errors there will be in the resulting material and thus how we guard ourselves toward embarking on an impossible procedure.

As a final remark it should be said, that there is of course the fact - although unpopular among cataloguers - that the cards used initially contains errors. We will in the following neglect this kind of errors and concentrate on the possible methods of correction of errors generated by the process itself.

2. Error types generated by the process

For larger catalogues it is important to control, that all cards are scanned. This requires a kind of bookkeeping, so that it is possible in every step of the production to keep track of the cards, and one good idea here might be as a start to use the kind of counting machine used in processing of checks to count the cards. In this way it is possible to tally the cards all the way through the process.

The next step is scanning of the cards and in this report we will suppose, that the process used for retroconversion starts with a scanning of the cards (or the bibliography) containing the source-material, after which the pictures of the source-material are processed by an OCR-package, producing the machine-readable records used as an input for the reformatting program.

When the OCR-process is finished, we have the input-material ready for the reformatting process, but probably this material will contain errors. It is important to realize, that for example to proofread all the produced records will in practice be impossible. So it is mandatory to be able to detect and correct errors from this part of the process efficiently.

The reformatting processes start with a parsing-operation, where the record is analyzed and broken down into its parts, e.g., title-statement, authorship-statement, shelfmark, printing place, notes etc. Parsing is done according to the formal description of the parts of the record; in a way this formal description is another expression of the cataloguing rules and practice used in generating the source material. So to achieve a result of good quality, it is necessary to have an input material of good quality.

Theoretically it should be so, that if the records from the OCR-process were perfect and the description of the records perfect, then the reformatting should also be perfect and thus should not generate errors. So to control the errors in the resultant process, it is very important first to control the errors from

the OCR-phase and next to have a perfect description of the cards. But first we will look on the OCR-phase.

3. Types of OCR-generated errors

Statistical analysis can reveal the patterns of these errors. There have been made several studies of database-quality using statistical frequency-analysis, the largest and best being the analysis of the Chemical Abstract's database [5]. The study reveals, that it is possible to describe most of the errors as substitutions between two characters. Many of these errors have their origin in normal typing-errors.

Similarly most of the OCR-errors can - as seen from f.ex.[7] - be seen as substitutions between two single characters.

This is following our experiences. There are of course in the material what might be called standard-errors, as it is not possible for an OCR-package to discriminate between two characters, if the graphical pictures of these characters are the same. So 1 (one) and l (small l) will be confused. Also there will be errors, where the pictures of two characters are close, for example 5 and z, a and e, 0 and 0(zero). Accents of every kind are another weak spot in the OCR-process. Furthermore most of the packages have difficulties to discriminate between upper case and lower case for characters with the same graphical picture. Yet another common error is the difficulty to reproduce the spacing between the words, especially when a proportional font is used in the original material. So an effort to correct these well-known typical errors should be done as a first step.

But apart from these kinds of errors, relatively few substitutions typically will account for a substantial part of the errors in the converted material.

The experience with this project is, that "good" material after the OCR-recognition and before the correction has an error rate of less than five pr. 1000 recognized characters and "fine" material has an error rate of .5 errors pr. 1000 characters. A characteristic for the errors is also, that for "fine" material very few errors make up for a very large percentage of all errors. Another experience is, that to estimate this central error rate it is

necessary to make an actual test-run of the cards, it is not possible just to have a look of the cards as the eye is not reliably able to estimate, if a material is suitable for OCR-conversion; but the sample need only to be of size around fifty cards to give a reliable estimate of the errors.

Normally an OCR-package will recognize a given character with a certain probability, and if this probability is too small, the OCR-package will mark the character as uncertain. It is important to note, however, that even after a correction of the marked errors, the converted text may still contain errors generated by the OCR-recognition. Some OCR-packages contain modules, which correct the most obvious errors of this type, but the most efficient packages contain a spelling-checker, which checks the final text. The weakness of this approach is, that this spelling-checker can only check one language in a document, and the material here is multilingual. So there is a need for at least a modified approach to make corrections of this material. So it might be worthwhile to take a closer look on the construction of spelling checkers.

3.1 Spelling checkers.

A blueprint of a spelling-checker has been given by Jon Bentley [1], where the construction is given as:

This blueprint is intended for a batch-process, but the principle given is valid also for a modern spelling-checker. The main operations done are

isolation of the words, conversion of capitals and stemming of the words followed by use of a dictionary. The dictionary is compressed as much as possible by using very elaborate and shrewd ways of storing the dictionary in main memory, because it has

not in the past been possible to make the spelling checker run in real-time using a disk file. This requirement is also the main reason for use of the stemming process.

A recent survey of spelling checking and correction is given by Karen Kukich [3] in a very large paper from 1992. The survey distinguishes between checking and automatic correction and considers of course the last job as the most difficult.

According to this survey, spelling checkers are still made from the same basic components as mentioned by Jon Bentley. It has not been possible to include linguistic knowledge in the process and more advanced methods like the use of neural networks have failed in practice, mainly caused by extreme demands on computer-power even for modest number of words in the glossary to correct. So there is one central component of all spelling checkers, namely the database or dictionary to use for comparison of the words.

As mentioned above it has not been possible to construct a database-organization allowing to store this database on disk, if the checker should run at a reasonable speed in a real-time process. Therefore there has been made considerable effort to make this database small and compact to keep it in RAM, resulting in very refined and complicated ways of storing the data. The most frequent used is to store the words as trigrams, i.e., overlapping combinations of three letters.

Automatic correction of errors is successfully done only for applications with a very limited vocabulary, i.e., less than 100 words. For an application like this one, it is possible to make simple correction-routines for example using the so-called reverse-error method, where the knowledge of the statistical pattern of the errors is used to calculate a probably correct word from a word not in a dictionary.

In project FACIT we have the goal to use ordinary PC's and if we look at a simple record, there will be roughly one hundred words to check in it. With the pertinent disk-technology, it will last around 20 milliseconds to look-up each word in a normal disk-file, making the response time for a check to be about two seconds, so it will be necessary to hold the dictionary-files in core.

If we assume a normal PC with 8 Megabytes of RAM, there will be the order of 7 Megabytes left to data. If we assume that one language requires about 60.000

words to represent the basic words, each of these words are of a length of eight characters and there will be six variants of each word stored, the immediate need for RAM is about 2.880.000 characters for one language, making it possible to work with two languages simultaneously.

After the survey by Karen Kukich has published, there has been described a new method to store words in a highly compact database in core, see ref.[6] and a single application of it has been given, apparently with success see ref. [4]. The compression here attained is reported to be in a size of eight, so it should be possible to use this organization and work with at least up to six languages simultaneously, which should be sufficient for most applications. Furthermore the established database will have the property, that the time spent in the database to check if a word is present in the database, will be proportional to the length of the word, not dependant of the actual size of the database. So this datastructure will be a convenient way to store the words to check.

Another help in accomplishing the goal of compactness has been to strip the words, so that it only is necessary to keep the roots of the words in the database. This approach is highly language-dependant and not suitable for for example Portuguese - and probably other Roman languages -, as there will be too many nonsense-roots produced using this method, see ref. [4]. So if a spelling checker has to contain many languages, the stemming-routine has to be left out and the database should contain all forms of the words used for checking.

To conclude, it is possible to construct a multilingual dictionary to check the words after the OCR-process and to attain the necessary speed in the process by containing the dictionary-files in RAM.

The data-structure might be the one mentioned above storing whole words - or trigrams generated from the words used to form the dictionary.

3.2 Possibilities of correction.

As mentioned above, a central issue in retroconversion is to be at least partially able to control the error-rate, errors here defined as aberrations from the original text produced by the

retroconversion process as a whole. When the word partially is used here it means, that for some errors it will be a waste of time (and money) to seek complete control, but for the central fields it will be mandatory. So to control the errors is a selective and individual process.

The reformatting process might also be seen as an enrichment of the original data by the inherent information from the format of the card, so that the resultant fields contain well-defined information, which might be easier to check. An example is the authors-field, where it of course is easier to control, that this field really contains a name than to control the whole card. This philosophy means, that it will be necessary to control the errors early in the process to the extent, that reformatting at all is possible. So for example one will prefer if possible to correct the standard-errors in an early stage of the process and to use dictionaries of various kinds in a later stage to make a general control of for example names and title statements.

An issue in this is the word-lists used to generate the dictionaries. It is possible to get lists of names from several sources, and of course it is also possible to make a formal authority-control of author names. But when it comes to words of a given language, there are some difficulties.

First it is a necessity to be able to decide, which language we should correct, as correction will be inefficient if the language is not known in advance. Furthermore there is the question of size of the dictionary and the problem of making the words in it reflect the time and the scope of the material converted. There will be a difference between converting and correcting material from an early twentieth century collection from a general library in Denmark and a collection mainly containing material from the eighteenth century from an Italian library.

It is well known, that there are several sets of word lists available for many different languages and it will be necessary to analyze them to obtain an approximate coverage of a language. The open question in this is, when are enough enough? The problem is, that the larger proportion of the words encountered in retroconverting the catalog of a research library or special library probably should not be found in an ordinary word-list or even a normal commercial dictionary, so in all circumstances it will be

necessary to have an easy way to augment the word-lists used for spelling and/or correction. There are though some indications, that word-lists might be comparative small, see ref. [8]. We have estimated a size of 60.000 words.

Word-lists from known languages also may be used to make an estimate of the language used in the text. In another research-project the use of neural networks for this purpose has been investigated. The result is very encouraging, as the network can decide the language of the title with a precision of app. 97 percent, discriminating among the five languages Danish, English, German, French and Spanish, see ref. [2]. The word-list used has been a list of titles from the State and University Library on-line catalogue with a known language code.

Practical experience has shown, however, that the language code has a very good correlation to the place of print; for typical retroconversion of older material the correlation will be better than 95 percent.

Combining these two measures, it is possible to obtain an estimate of the language of the publication with very good precision and thus it is possible to use a language-specific dictionary to check the title words. As mentioned above, the words used for the dictionary will be critical, because they have to represent the glossary found in the texts converted, and the dictionaries could be either words or trigrams; we think at this stage, that trigrams have a slight advantage, as they could represent words not encountered in the basic word-lists, thus they might be more stable and need less updating in the process.

So a tentative conclusion is, that it is possible to make an error-detection for selected fields, given the assumption that suitable word-lists may be found. Simple routines for error-correction can be constructed using the mentioned reverse-error method, after a study of the normal statistical patterns of error in the converted material.

3.3 Outline of the method.

The method could be outlined as follows:

1. Find suitable - not too large - word lists from f.ex. Internet or a library-catalog for the relevant languages.

2. As a prelude, make a thorough analysis of f.ex. the first fifty records processed by OCR to get an impression of the pattern of errors for this catalog and this OCR-package, i.e., find the errors produced and make a table of the most often encountered transformations from the errors.

3. After the fields suitable to correct has been isolated, make the corrections after the following guidelines:

a. The parser suggests the language from the place of printing. An analysis of the text-field chosen confirms the assumption or issues a warning, after which the operator chooses the relevant language.

b. Every word in the field considered is for this purpose converted to lower-case letters.

c. If the word is a number with or without comma/point, it is accepted and the next word is considered.

d. If the word is in the dictionary, it is accepted and the next word is considered.

e. If the word is not in the dictionary, list suggested words using the list of typical errors backwards. Reduce the list by checking it against the dictionary.

f. The operator makes the last choice and she/he has the following options:

- Accept one of the suggested corrections, if there is any suggestion made

- Correct the word in the text and add it to the word list

- Correct the word in the text, not adding it to the word list.

The procedures described above could be used as a general procedure for all table-look-ups in the FACIT system, but it should be designed with the goal to be an interactive and supportive procedure, not an automatic correction procedure.

It is also assumed, that the original picture of the card is readily available for the operator throughout the process as in most OCR-systems, when using the built-in facilities for error-correction.

4. The reformatting process

As said above, the reformatting process itself depends on a formal description of the catalogue cards, this formal description being another expression of the cataloguing rules and the practice used during the creation of the catalogue.

This means again, that if this practice has varied a lot - making the catalogue inhomogenous - it will be difficult to create one description covering all instances of the cards.

WE think that old catalogues generally are inhomogenous and so it might be preferable not to make a detailed reformatting operation, but rather make the target OPAC-system compensate for the missing opportunities due to the crude formatting.

Another snag in the reformatting is missing tokens. The reformatting may have for example to detect a token for a field separator. It might be a point, a special character or an extra blank line, or the simple fact, that the authorship-statement always is contained in the first line of the card and always only is one line. If one builds a description around the last mentioned fact, that the authorship fills exactly one line, then there is a real risk, that the reformatting sometimes will produce an error, as many OCR-packages of today tend to generate an extra line-shift randomly, and when this happens in the first line, the resultant formatting will be wrong. Consequently, it is necessary to make a control afterwards. It might be as simple as to make and browse a list of all author-fields found in the cards or to make a check of forenames and surnames or an authority control.

It is our experience, that it is advantageous to keep the reformatting as simple as possible, producing it in a top-down development process and keeping complicated expression out of it.

The consequence of this is, that the development of a description is a highly iterative process, compromising the ambitions of making a detailed

reformatting and making the description cover as much as possible of the catalogue. Thus, the formal description of the catalogue should be done by a person with a good working knowledge of both the target OPAC-system, the catalogue and the shifting practice of it. It is not a routine matter.

5. Recommendations

In summary, we will make the following recommendations concerning error-control:

1. Before start of the process, test the actual OCR-process using a representative sample of the catalogue of fifty cards.

If the raw error-rate is greater than five errors per 1000 characters read, do not embark on the project!

If the error-estimate indicates, that it is reasonable to embark on the project, make a distribution of the errors to be used later in the correction process.

2. Use the OCR-systems built-in facilities to correct the obvious errors.

3. Make an error-control system by either simply detecting the error-prone combinations or making a small program using trigrams to detect the errors in the specific fields.

4. Install a tally-system to make sure, that all cards go through all steps of the process.

5. If possible, control the names, etc. by means of a dictionary. If this is not possible, make a skilled cataloguer browse lists of all authors, serial titles etc. to correct the obvious errors. This might also be a SCAN-operation, using the normal OPAC-terminals.

Last we think, that error-control is a continuing effort. So in practice the whole retroconversion effort should be iterative, so that after a certain batch of cards has been processed, the quality of the resultant records should be evaluated and the possible errors accounted for. The effort should be continued with samples drawn from the process at regular intervals.

6. References

1.
Bentley, J.
A spelling checker. CACM, Vol. 28, no. 5, May 1985,
p. 456-462

2.
Hørning, Annette.
Language Identification using an Artificial Neural
Net. Report, Department of Lexicography and
Computational Linguistics The Aarhus School of
Business, 1995.

3.
Kukich, K.
Techniques for automatically correcting words in
text. ACM Computing Surveys, Vol. 24, no. 4, Dec.
1992, p. 377-439.

4.
Lucchesi, C. L. and Kowaltowski, T.
Application of finite automata representing large
vocabularies. Software - Practice and Experience,
Vol. 23, no. 1, Jan. 1993, p. 15-30.

5.
Pollock, J. J. and Zamora, A.
Automatic spelling correction in scientific and
scholarly text. CACM, Vol 27, No. 4, Apr. 1984, p.
358-368.

6.
Revuz, D.
Minimisation of acyclic deterministic automata in
linear time. Theoretical Computer Science 92, 1992,
p. 181-189.

7.
Takahashi, H. , Itoh, N. , Amano, T. , Yamashita, A.
A spelling correction method and its application to
an OCR system. Pattern Recognition, Vol. 23, no. 3/4,
1990, p. 363-377.

8.

Zobel, J. and Dart, P.

Finding approximate matches in large lexicons.

Software - Practice and Experience. Vol. 25, no. 2,
1995, p. 331 - 45.

Appendix 1

Terminology and Basic Concepts

Source: The typewritten or printed source of the conversion proces; the "original".

Result: The machinereadable file in text-form, that is the result of scanning the source and applying the OCR-package.

Image: The digitized image, that is the intermediate result of scanning the source.

Unrecognized character: Character not recognized by the OCR-package and marked with a special character (i.e. * or @)

Misrecognized character: Character recognized by the OCR-package as another character.

Doubtful character: Character recognized by the OCR-package, but marked as doubtful (usually because it is below a specified level of uncertainty. (Alternative: Uncertain character)

Untrained character: A character in the source that has not been presented to the OCR-package during the training. Such a character will either result in an unrecognized character or a misrecognized character. Only with OCR-packages with Omnifont facilities and the like is there a possibility of getting a correctly converted character and only if the Omnifont character set includes this character.

Missing character: Character present in the source but not represented in the result, not even as a SPACE-character.

Inserted character: Character not present in the source, but present in the result.

Missing SPACE: SPACE-character present in the source but not in the result, so that two sequences of NON-SPACE-characters separated by one or more

SPACE-characters are contracted into one sequence of NON-SPACE-characters.

Inserted SPACE: A SPACE-character present in the result, making what is one sequence of NON-SPACE-characters in the source into two sequences separated by one or more NON-SPACE-characters.

Alphabet: A set of characters in a specific form, like Latin characters, Greek characters, Cyrillic characters etc.

Character set: A full set of characters (letters, digits, diacritics/accents, punctuation marks and other characters) in one alphabet, disregarding size, typestyle and typeface (font).

Repertoire of characters: The set of characters actually occurring in a given catalogue. It is important to make sure that the OCR-package is able to handle the whole repertoire of characters, while characters from the full character set(s) not occurring may be ignored, i.e. in training the OCR-package.

Typeface (font): Characters in one alphabet but with a specific form, i.e. Courier, Elite, Pica, Letter Gothic, Times Roman, Dutch, Helvetica etc. Characters in the same typeface may have different sizes and different typestyles.

Typestyle: Characters in the same typeface, but with a specific form: i.e. ordinary, bold, italic, underlined.

Size of type: Characters in the same typeface but with a specific size.

Basic character: Character from a given character set, that may be used on its own and is not a composite character.

Composite character: A character that may be analysed into two or more separate components that may occur alone or in combination with other characters, i.e. a basic character with diacritics/accents.

Diacritic(al mark): Any character or mark, used to differentiate the phonological "meaning" of a basic character: a å ä á â â, n ñ, ç.

Ligature: Use touching characters.

Touching characters: Characters in the source that are touching each other or overlapping in such a way that the OCR-package is not able to identify the individual characters. (Used for ligature).

Broken characters: Characters in the source that have white space going through what should have been unbroken ink/print.

Split characters: Characters in the source that are split vertically by white space.

Misprint: One or more characters in the source representing unauthorized spelling or deviations from the form in the original work. Could be missing characters, inserted characters substitution of characters, and characters in the wrong sequence, as well as combinations of these.

Letter: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, æ, ø, å, and composite characters based on these. (And the equivalent in Greek, Cyrillic etc.)

Digit: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0

Punctuation mark: . : ; , ! ? -

Digram: Sequence of two characters.

Trigram: Sequence of three characters.

N-gram: Sequence of N characters.

Error rate: Number of misrecognized characters per 100 characters in the source.

Accuracy: Number of correctly recognized characters per 100 characters in the source.

Speed: Number of characters (in the source) processed per time unit, excluding or including time to handle the source. Number of cards processed per time unit, excluding or including time to handle the source.

Noise: Any spot, smudge or other marking in the source that the OCR-package will try to interpret as a character or a sequence of characters. The whole at the bottom of some cards may be treated as noise in this respect.

Probability of conversion: The probability that a specified character or a sequence of characters in the source will be converted into a specified character or sequence of characters (including the NULL-character).

Transition table: A table showing correspondances between characters (or sequences of characters) in the source, and characters (or sequences of characters) in the result, with the probability of transition. The probability of transition may be given with the source as the point of origin or the result as the point of origin, or both.

Example:	1	60	45	1
	1	30	35	l
	1	10	10	I

This means that a 1 (one) in the source will be converted into a 1 (one) in 60 pct. of the cases, into an l (el) in 30 pct. of the cases and into an I (capital i) in 10 pct. of the cases. A 1 (one) in the result is a correct conversion of a 1 (one) in 45 pct. of the cases etc.

Error-prone characters: Characters with a low probability of being correctly converted.

Environment of conversion: The scanner, contrast setting, OCR-package, state of training of the character set, and the source of conversion taken as a whole. Errors will have