ELECTRONIC PUBLISHING, VOL. 6(4), 435–446 (DECEMBER 1993)

Qualitative analysis of low-level logical structures[1]

ABDEL BELAÏD, JULIAN C. ANIGBOGU AND YANNICK CHENEVOY

CRIN-CNRS / INRIA Lorraine
Campus Scientifique ; Baî. LORIA B.P. 239
54506 Vandoe uvre-le's-Nancy Cedex
France

email: abelaid@loria.fr

SUMMARY

This paper presents a qualitative approach to logical structure recognition of library references. The system is driven by a generic model of a reference class and by an OCR **flow, given in** SGML **format, that include** ASCII **code of the characters and information about the typographic style and the lexical affiliation of words. The approach used is based on hypotheses production and verification about the existence of sub-field limits in the reference area. At each step of the analysis, the generated hypotheses are sorted on the basis of their confidence scores and the most likely hypothesis is analyzed. The result is a structured flow containing, in** UNIMARC **format, the list of different sub-fields recognized, accompanied with their confidence score.**

KEY WORDS Document analysis Low-level logical structure Qualitative analysis ODA formalism Library reference

1 INTRODUCTION

Document analysis has become, with the progress in OCR technology and electronic publishing, a helpful tool for document input. Many attempts in this area have been made in the last decade [1–4]. The main approaches tend to use a domain-knowledge model to segment the document image into homogeneous regions and to label them according to the model description. An optical character reader is used to extract and recognize the text components. The challenge of such systems is to retrieve the document content in spite of the possibly bad quality of images and the uncertainty of the generic model, and to incorporate text reading into structure analysis.

The key concept in document analysis is that of structure [5,6]. It gives the information to guide the system's actions. Often, document analysis consists in finding the high-level structure in the document in terms of a hierarchy of physical and logical objects. At this level, the layout components, dealing more with the visual perception of the document, correspond to big regions such as *columns*, *text blocks*, *graphics* or *photographs*. The *logical* components, more connected with content interpretation, give for instance, *titles*, *paragraphs*, or *footnotes*. For this high-level structure, the strategy would need to be guided by the physical aspect which is rich and allows the easy location of the different components [4,7].

[1] This work was funded by the EEC libraries programme LIB-MORE

CCC 0894–3982/93/040435–12

Received 15 August 1993
Revised 1 December 1993

There are many applications where it is necessary to pursue the analysis across the content and to extract a finer structure. This concerns the content architecture of physical blocks such as *footnotes*, *figure captions*, *bibliography references*, etc. In these cases, the physical structure is generally limited to a sequence of lines and words, not very much exploitable by the analysis. The logical structure should be more informative. It provides beyond field labels, a fine description of the content such as the typographic style, the linguistic affiliation of words and contextual information on successive sub-fields such as separators and their location. Document analysis here resembles document understanding. Several document image understanding systems have been proposed [7,8] for documents such as letter addresses, forms, or library references where the *logical* structure is more complete than *layout* structure.

This paper describes structure modeling for low-level logical structure analysis of library references and its inherent recognition strategy. The terminology is based on the ODA international standard. Because of the uncertainty of the OCR data flow, we use a qualitative analysis based on hypothesis evaluation. This allows to take into account what is important to recognize and to evaluate the solution retained.

2 LOW-LEVEL STRUCTURE DESCRIPTION

2.1 Reference description

References are typed in several library catalogues and sorted by month and year. Figure 1 shows an example of a library reference. The physical structure is partitioned into five areas. The first area, composed of the first line, contains, on the right hand side, the 'CDU' code (Classification Décimale Universelle) which gives some information about the library classification of the reference. The second area contains the reference body composed of a series of fields describing the referred work in the reference such as: '*vedette*' (author name or beginning of title), '*title*', '*address*', '*collation*' (material description of the work: location, editor, year, format, etc.). The body is often typed in many lines. The third area contains the '*Collection*' field (description of the series, volume, etc.). The fourth area contains the '*Note*' field which gives information about, for example, the title (abbreviated, complete, original, etc.). These last two areas are optional and so are not always present in the references. The last area, located on the last line of the reference, contains the '*Reference*' on the left hand and the '*Order number*' on the right hand.

2.2 Structure modeling

Knowing that the problem is to find the sub-fields within reference areas, the model specification concentrated on the description of sub-field properties, by the distinction of their typographic styles, the existence of particular words or group of words and their appearance in certain lexicons, and essentially their limits (type of initials and finals such as capital letters, particular words, or type of punctuation separating the sub-fields).

The ODA formalism was used to model the reference structure. The model is given by a context-free grammar written in the EBNF formalism. The format of a production rule is as follows:

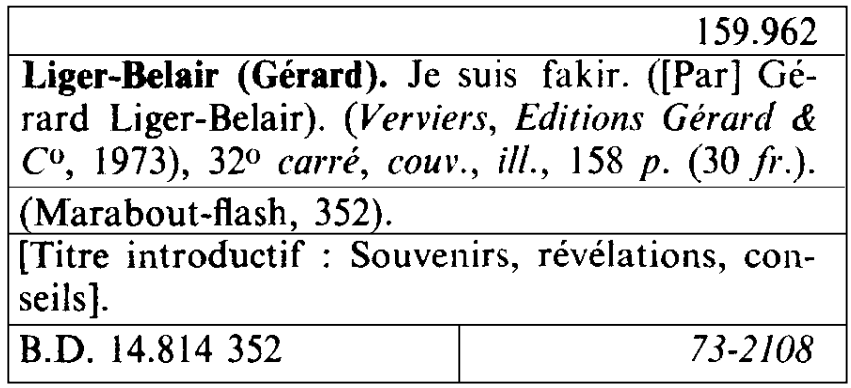

Figure 1. Example of a library reference

$$\begin{array}{lll} \textit{Term} & ::= & \textit{Constructor subordinate\_Objects}[\textit{Qualifier}] \mid \\ & & \textit{Constant} \mid \textit{Terminal} \\ \textit{Constructor} & ::= & \textit{seq\_td} \mid \textit{seq\_lr} \mid \textit{seq} \mid \textit{aggr} \mid \textit{cho} \mid \textit{import} \end{array}$$

2.2.1 Constructors and qualifiers

A term, the left hand of a rule, can be simple (constant or terminal) or composed of subordinate objects. In the latter case, a constructor describes the relationship between objects in the term area. The constructor precises the order of the appearance of subordinate objects such as SEQUENCE: top-down (*seq_td*), left-right (*seq_lr*), or logical (*seq*), *aggregate* (*aggr*), or *choice* (*cho*). A special constructor '*import*' is used to inherit for the term some or the total description of another existent and similar term. Furthermore, to express the object occurrence in the term, each object may be accompanied by a qualifier such as *optional* (?), *repetitive* (+), or *optional-repetitive* (*). Here is an example:

REFERENCE	::=	*seq_td* CDU GLOBAL_BODY ZREF
GLOBAL_BODY	::=	*seq_td* BODY ZCOLLECT? ZNOTE*
ZREF	::=	*seq* REF ORDERNUMBER
...		

The term 'GLOBAL_BODY' is described as a top-down sequence of terms where 'ZCOLLECT' is optional and the note area 'ZNOTE' is optional-repetitive.

2.3 Attributes

Because of the weakness of the physical structure and the multitude of choices represented in the model, we add to the previous description some attributes given by the library specification to refine the description of the reference components. These attributes are associated to each rule in the following format:

$$\textit{Attribute\_name} \quad \{[-]\textit{value}[\textit{weight}]\}*$$

The minus sign indicates the negation of an attribute value. Several kinds of attributes have been defined, among them, *Type* (string, line, word, char, etc.), *Mode* (capital, numeric, alphabetic, punctuation, etc.), *Style* (bold, italic, standard, etc.), *Position* (beginning of line, inside, end), *Lexicon* affiliation (author index, countries, towns, abbreviations, articles, etc.), *Separator* between subordinate objects (space, comma, hyphen, etc.), *Weight* which specifies the degree of importance of subordinate objects, etc.

Example:

TITLE	::=	*seq* PROPERTITLE RESTOFTITLE
Style		-italic
Position		Begline
Sep		Comma

This example describes the term 'TITLE' as a logical sequence of two objects: 'PROPERTITLE' and 'RESTOFTITLE' where the style is not italic (may be bold or standard), which is located at the beginning of the line and whose separator is a comma.

2.4 Weights

Given an uncertain OCR flow and in order to obtain an evaluation of the retained solution, weights are used. These weights are specified in symbolic form, for example [A,G], and the actual numeric values are determined from a base value specified by the user. In this manner, the user can specify the importance that he attaches to each subordinate object.

Example:

ZPB	::=	*seq* LCAP RP PARTICULE?
Weight		PARTICULE A LCAP G

The optional object 'PARTICULE' (A) is more important than 'LCAP' (G). This specification is logical since an optional object normally helps in reinforcing the possible presence of a term more than an object that is always present.

2.5 Remedial actions

The processing of a rule sometimes needs a check procedure before or after its processing, either to prepare the analysis of the rule, to recover from failure of the analysis or to post-process the result. For example, if the analyzer is expecting only numeric values in a region being analyzed but fails to find them, and if there is a remedial user function attached, it is called (usually this function should check for OCR substitution errors, ex: 'l' for '1', 'O' for '0').

Example :

FIP	::=	*cho* FIP1 IPA
Clex		-Abn
Style		italic
Action		+VerifyStringInField(fr.,false) Restitute(215,bb)

This production rule describes a choice between two terms neither of which should contain any of the strings in the lexicon Abn (expressed by the attribute Clex). There are two actions. The first one indicates to verify before the rule analysis that the search zone does not contain the string 'fr.'. In the event of the hypothesis being verified, the second function is executed to create a UNIMARC tag before restituting the result in the required format.

3 STRUCTURE ANALYSIS: THE PROBLEM DATA

Several problems arise while trying to extract the logical structure from a document. The major one is that the image represents the physical structure, while the recognition objective is to identify the logical components. For this, we have to resolve two sub-problems:

- find the physical structure, i.e., to cut the document image into homogeneous and structural entities: blocks, lines, words, etc.
- convert the physical structure into a logical one, i.e., to retrieve the document 'semantics'.

Because of the finesse of the reference structures, the solution to these two problems is searched for with a lot of precaution. The reference sub-fields are characterized by very unstable and fragile indications. For example, the typographic style such as bold or italic which is capital for the location of the author name and the title, is often difficult to determine by OCR. The style either overflows into several sub-fields or falls short of the entire sub-field area. Punctuations which mark the limit between sub-fields can sometimes disappear or be substituted with something else. Word recognition does not always succeed, which leads to lexical affiliation errors. Furthermore, a lot of the sub-fields are optional, and as such complicate the system's actions because it has to take into account all the possibilities.

An approach to this kind of data analysis consists in constructing all possible segmentation hypotheses of sub-fields from all the possible separators and to evaluate them in order to find the most likely combination and order of sub-fields. This is to reduce the system errors and ambiguities.

4 SYSTEM OVERVIEW

Figure 2 shows the major components of the developed system. The *Input* is given by tagged OCR output references. The *Domain knowledge* contains the generic structure of the references (model) and also the data flow to recognize. The flow is given in SGML formalism. The model is compiled and the flow is filtered to extract interesting information for the analysis.

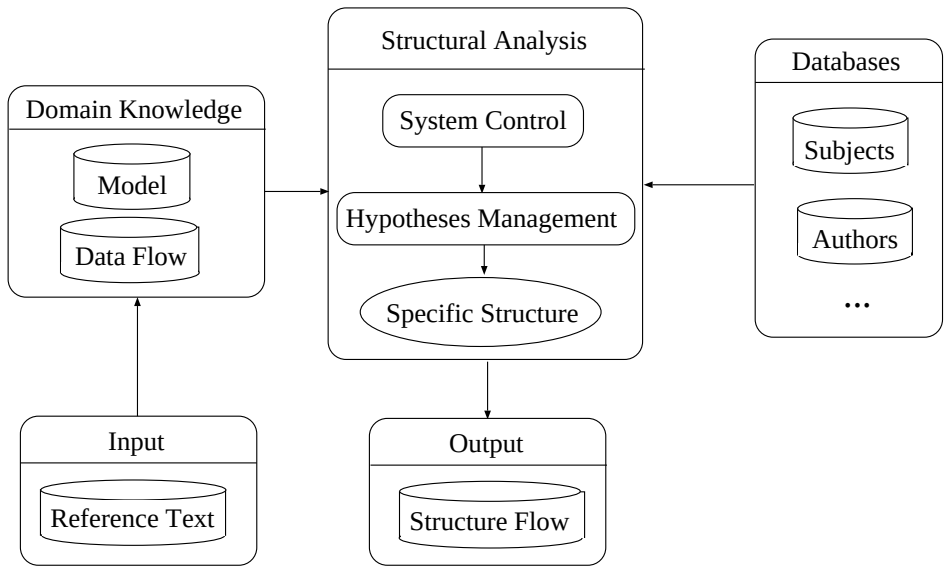

Figure 2. System overview

The *Databases* contain all lexicons used for term verification (index of author names and titles, towns, countries, expressions, etc.). These indexes were recognized in a previous step in the same manner, that is, with rules and tagged OCR output. What follows, describes only the reference analysis.

The *Structural analysis* is the main module of the system. The system control acts on the hypotheses management process to produce and verify hypotheses on the specific structure. The specific structure is represented by a tree of terms. Hypotheses are produced for each node of the tree and construction is pursued first for the node which looks most likely (has the highest score).

The *Output*, tagged in UNIMARC, contains the specific structure identified by the system.

5 INPUT DATA

The data flow is obtained from a layout analysis of the catalogue page images and a reading text process.

The result of these tasks is a data flow containing the reference text coded in SGML. The tags separate the lines and different information such as style or lexical class corresponding to each word (token). Figure 3 shows the flow corresponding to the reference of Figure 1.

Each reference is located in this flow between two successive tags ‘<NOT>’ and ‘</NOT>’. Useful tags for the document analysis are ‘LEX’ which gives the lexicon affiliations of words, ‘I’ for italic style, ‘B’ for bold style, and ‘S’ for the number of spaces. The default style is standard and as such not tagged.

```
<DOC TY=N PROV=ENRLEX EG=OK NPN=2085 NDN=2114 IMA=users/brb/juin73/images>
<PAG NP=1 NOM=0008.ima> <COL XHG=63 YHG=1900 XBD=1027 YBD=2912>
<NOT NON=2108> 159.962 <LEX L=AFR><B>Liger-Belair <LEX L=GNL,AFR><RED
F=253>(G{\'e}rard).</B> <LEX L=GFR,GNL>Je <LEX L=GFR>suis <LEX
L=GFR>fakir. <LEX L=GFR,GNL>([Par] G{\'e}rard <LEX
L=AFR>Liger-Belair). <I>(Verviers, <RED F=253>Editions <LEX
L=GNL,AFR><RED F=253>G{\'e}rard \&</I> <LEX L=GFR,GNL>C, 1973),
32 <LEX L=GFR,GNL,AFR><I>carr{\'e}, <RED F=253>couv., <LEX
L=GFR,GNL>i11.,</I> 158 <LEX L=GFR,GNL><RED F=253>p. (30 <LEX
L=GNL><RED F=253>fr.). <LEX L=GFR,GNL>(Marabout-flash, 352). <LEX
L=GFR>[Titre <LEX L=GFR>introductif: <LEX L=GFR>Souvenirs, <LEX
L=GFR>r{\'e}v{\'e}lations, <LEX L=GFR,GNL>conseils]. <LEX
L=GFR,GNL><RED F=253>B.D. 14.814 352 <S N=14><I>73-2108</I>
</NOT>
</DOC>
```

Figure 3. Flow of the given reference

6 DOMAIN KNOWLEDGE

The information given by the model and the data flow is reorganized to be used by the analysis process.

6.1 Model compilation

Grammar text is converted by a syntax analyzer and code generator into a working structure. The working structure is a dynamic table of terms where the entries correspond to term codes. Each term is given by a list of characteristics gathered in a characteristic table. This allows the system to read rapidly the characteristics of each term analyzed.

6.2 Data filtering

The filtering task consists in extracting the useful information for structural analysis. In order to do this, the system generates two structures:

- a buffer containing only the text (without the tags),
- a table which contains useful tokens extracted from the flow such as style, token size, etc., and a pointer to the buffer.

7 STRUCTURAL ANALYSIS

7.1 Principle

The objective of our approach is to introduce qualitative ‘reasoning’ as a function of the recognition evaluation. This evaluation allows:

- the reduction of errors and ambiguities due to faulty data (OCR errors, data not fitting the model specification, etc.),
- taking into account what is important to recognize,
- the qualitative evaluation of the obtained solutions,
- the isolation and separation of doubtful areas.

To achieve this objective, weights are first assigned by the user to the terms and attributes according to the degree of importance he attaches to them. Objects for which weights are not specified are automatically assigned weights as a function of the number of objects present for this term as well as the symbolic weights already assigned to the other objects by the user.

For example, the 'CDU' is of type 'line', as it always fits into a single line (this automatically means that the analyzer will conclude that it must end with a carriage return), alignment 'right', mode 'printable', as any character can be in this zone. The style is standard (default style and so need not be specified). Weights are attached to attributes that the user feels are important for good recognition of the field or object (left-hand term). To stop an attribute from playing any role, the user attaches a weight of zero.

The expressive power of our grammar is such that an object can have several values of the same attribute. For example, if the user has no more information about the field than that it is non-numeric and does not contain punctuation, this translates into something like:

Mode -num 5 -punct 3

where 5 and 3 are relative weights attached to non-numeric and non-punctuation, respectively. These weights, just as for subordinate objects, are optional.

The specific structure is a tree of nodes corresponding to the terms analyzed. At each step, the system further decomposes terms stacked in an agenda. This stack is sorted by decreasing order of *a priori* scores. Since the agenda is always sorted in decreasing order of hypothesis scores, the analyzer is said to function in an opportunistic mode. That is, it always first selects the term that looks most promising. Thus, it can move from one branch of the hypotheses tree to another in no 'apparent' order. Terms that are no longer decomposable (i.e. leaves) are directly verified and as such either pass or fail. If the analysis fails and a recovery function is present, it is first executed before the next item on the agenda is selected and processed. This process continues until the agenda is empty. The result and evaluation scores for the different nodes in the hypotheses tree traversed to produce the output data for the fields that need to be restituted. For example, in the references we treated, names are transformed from (Surname Firstname(s)) to (Firstname(s) Surname) while addresses are not restored.

The different analysis steps are detailed below.

7.2 Initialization

1. In the beginning, the agenda is initialized to the model axiom.

   (Code for REFERENCE, 100)

2. The specific structure is then created by initializing it with the one current hypothesis (see Figure 4)

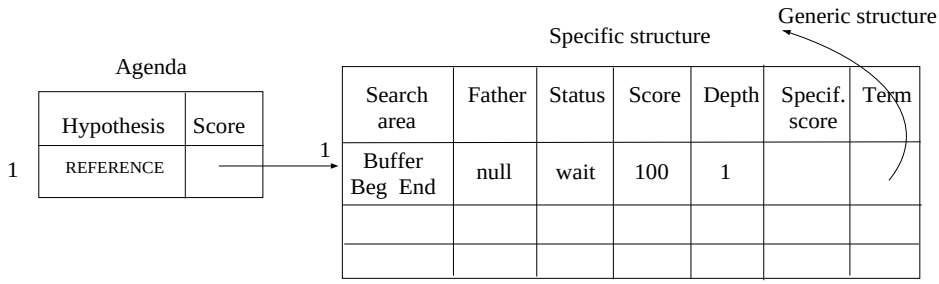

Figure 4. Schema of the specific structure

Status :

wait : hypothesis not yet verified
passed : hypothesis verified with success
failed : hypothesis failed

Score : *a priori* hypothesis score based on the verification or not of the attributes, the initials and finals of the generic term being verified.
Term : pointer to the generic term.
Father : pointer to the generator of this hypothesis (null for the root).
Weight : the more an object score is important, the more its evaluation will have an influence on the final score. Weights should be big when we approach the root. Inversely, the more we go down in the tree, the more weights become less and less important.
The weight is automatically computed during the analysis. It depends on the depth of the father node (in the specific structure) and on the symbolic weight given by the user as well as the scores of the attributes verified.
Let the rule be:

$$O_p :: = seq\ O_1\ O_2 *\ O_3 ? \ldots O_N$$

where $O_i \in \{O_1, O_2, \ldots, O_N\}$ is a subordinate object and N is the number of subordinate objects for the term O_p.
The evaluation function E is given as:

$$E(O_p) = \frac{\sum_{i=1}^{N} P_{O_i} \times E(O_i) \times L(O_i)}{\sum_{i=1}^{N} P_{O_i} \times L(O_i)}$$

where P_{O_i} is the weight of O_i, $L(O_i)$ is the number of characters that make up subordinate object O_i, and N is the number of subordinate objects that compose the term. P_{O_i} is defined as:

$$P_{O_i} = p_{O_i} \times \frac{D}{d}$$

where:

d = depth of O_i in the tree,
D = maximum depth in the model (21 in the present case),
p_{O_i} = weight deduced from the symbolic value given by the user or computed as a function of the number of brothers.

In this manner, the contribution of a son to the final score of its father is also a function of the proportion of the area it occupies in the zone assigned for its father. Thus, correctly recognizing a subordinate object that is 20 characters wide in a zone of 25 characters (for the left-hand term, i.e. the father) has its importance.

7.3 Iteration

The analysis task is repeated while there are hypotheses to verify. The constraint imposed on the system is to find all solutions (not just the first, even if this sometimes leads to ambiguities). The procedure is as follows:

a) pop off the most likely hypothesis from the list of goals (agenda);
b) see if the hypothesis has already been processed in another context, in which case the result is directly inherited from the solution tree;
c) else, see if actions have to be verified before. If yes, perform them (there are actions that give directives to follow);
d) else (general case) analyze the constructor:

d.1) CHOICE
- the search area is the same as that of the father, in this case it is easy to search for frontiers;
- each of the choices inherits the *a priori* score of the left-hand term (father).
- if the confidence score is acceptable, we put in the agenda the corresponding subordinate object (the user can modify the cut-off score).

d.2) SEQUENCE, ex: Seq A B C

- find in the search area (zone of the left-hand term) all the initials and finals of the objects A, B, and C. The initials and finals are terminals given by the model (characters, words, etc.);
- construct all the potential zones in the buffer corresponding to the combination of initials and finals; that is, for each possible final of A, find an initial of B that is immediately to its right. For this combination, choose a final of B and find an initial of C that corresponds, etc.
- these potential combinations will be considered as hypotheses and stored in the specific structure. Each of the objects that is part of a hypothesis but is not a terminal is also pushed onto the agenda so as to be analyzed in its own turn.

d.3) AGGREGATE
- is simply a choice of all possible combinations of the subordinate objects. That is, all possible sequences (d.2) of the subordinate objects are constructed and treated as choices (d.1).

e) sort the agenda by score and return to b).

8 EXPERIMENTAL RESULTS

Figure 5 gives the analysis result of the reference given in Figure 1. All the sub-fields were correctly localized. They are coded and tagged in UNIMARC. 'QSTR' indicates the evaluation score (maximum 10 000).

```
<675 I=bb QSTR=10000> <$a QSTR=10000>159.962</$a> </675>
<200 I=0b QSTR=9834>  <$f QSTR=9487>Grard Liger-Belair</$f>
                      <$a QSTR=10000>Je suis fakir</$a> </200>
<700 I=b0 QSTR=10000> <$a QSTR=10000>Liger-Belair</$a>
                      <$b QSTR=10000>Grard</$b> </700>
<210 I=bb QSTR=9705>  <$a QSTR=10000>[Verviers]</$a>
                      <$c QSTR=9519>[Editions Grard & Cé]</$c>
                      <$d QSTR=10000>[1973]</$d> </210>
<215 I=bb QSTR=9750>  <$d QSTR=7353>32é carr</$d>
                      <$c QSTR=8601>couv., i11.</$c>
                      <$a QSTR=10000>158 p.</$a> </215>
<010 I=bb QSTR=10000> <$d QSTR=10000>30 BEF</$d> </010>
<225 I=2b QSTR=10000> <$a QSTR=10000>Marabout-flash</$a>
                      <$v QSTR=10000>352</$v> </225>
<517 I=0i1 QSTR=10000><$a QSTR=10000>Souvenirs, rvlations,
                      conseils</$a> </517>
<900 I=bb QSTR=9772>  <$a QSTR=10000>B.D. 14.814 352</$a>
                      <$b QSTR=9285>73-2108</$b> </900>
```

Figure 5. Structural analysis result of the given reference

The method has been tested on about 370 references. The average analysis time on a SPARC 10 is 20" with a minimum of 3" and a high of 3' depending on the complexity of the reference. This is the case when the reference does not fit into the definition of the model or the OCR made mistakes as regards, for example, style (which sometimes is the only information that can help identify a zone) or completely misrecognized characters or punctuations. Before the analyzer will give up on a reference, it is forced to explore all hypotheses it generated. Since the goal is to find all possible solutions, the fact that it encountered errors in critical areas does not mean that there are no solutions in other branches of the solution tree.

In the event of errors, the system generates a fictitious UNIMARC code 903 which it uses to demarcate the zone it should have recognized for a field, but which does not quite fit the characteristics as specified by the user. This helps in modifying the model to take care of exceptional cases or to really determine that the reference was badly formed as a result of OCR errors, the printers devil, or outright bad transcription of the reference.

When the system finds more than one solution for a given zone, it equally generates a fictitious UNIMARC code 902 that it puts around each of the possible solutions which are then presented to an operator who has to make a choice.

9 CONCLUSION

The system presented here gave good results on tested library references. The errors encountered were due to incomplete specification (reference not falling into any of the categories we were provided information on) or OCR errors. The ambiguities encountred were partly due to a combination of incoherence in the specification (which allows different legal segmentations) as well as OCR substitution errors. The evaluation allows the observation of the quality of each reference and each field in the reference and allows the user to intervene or not for manual correction.

REFERENCES

1. A. Dengel, 'Document Image Analysis', in *Proceedings of IAPR Workshop on Syntactic and Structural Pattern Recognition*, 78–87, (1990).
2. S.N. Srihari and G.W. Zack, 'Document Image Analysis', in *Proceedings of the Eighth International Conference on Pattern Recognition*, 434–436, IEEE Computer Society Press, Washington, DC, (1986).
3. K.Y. Wong, R.G. Casey, and F.M. Wahl, 'Document Analysis System', *IBM Journal Research Development*, **26**, 647–655, (1982).
4. Y. Chenevoy and A. Belaïd, 'Hypothesis management for structured document recognition', in *Proceedings of the First International Conference on Document Analysis and Recognition*, 121–129, (1991).
5. B. Brown, 'Standards for Structured Documents', *The Computer Journal*, **32**(6), (1989).
6. *Structured Documents*, eds., J. André, R. Furuta, and V. Quint, The Cambridge series on electronic publishing, 2, Cambridge University Press, Cambridge, UK, 1989.
7. Y. Chenevoy, *Reconnaissance structurelle de documents imprimés: études et réalisations*, Ph.D. dissertation, CRIN-University of Nancy, France, December 1992. (Thèse de Doctorat de l'INPL.)
8. S.N. Srihari, C.H. Wang, P.W. Palumbo, and J.J. Hall, 'Recognizing Address Blocks on Mail Pieces: Specialized Tools', *AI magazine*, 25–40, (1987).