

Information Retrieval Systems tools in Document Analysis and Recognition

A. David and A. Belaïd

LORIA, Campus Scientifique, B.P. 239
F-54506 Vandoeuvre-Lès-Nancy Cedex France
Email {Abdel.Belaid,Amos.David}@loria.fr

Keywords: Information retrieval systems, Objected oriented database, Content based information retrieval, Information analysis, Document analysis, Document formats

I. Introduction

We present in this paper how some techniques used in information retrieval systems (IRS) can be used to enhance the process of document analysis and recognition. Our studies have been centered on the types of query and result that can be obtained from an IRS. Our approach has been to represent knowledge on document structures in form of database. This allows the use of the IRS functions to access the knowledge on document structures. The fact that document analysis requires the employment of knowledge on document structures shows the potential contributions of IRS techniques.

In our studies, we represent two categories of knowledge on documents. The first category of knowledge concerns the structural organization of the class of documents to analyze. The knowledge in this category is occasionally available in a normalized form, such as BIB_TE_X for bibliographic references or SGML documents for books. The second category of knowledge concerns the physical structure of documents. This structure is generally associated to a printing style. The knowledge in this second category is also occasionally available such as the printing styles implemented in document generation systems.

In section II., we present the general architecture for document analysis in which the knowledge on documents is managed by an IRS. A brief description of the types of result and the types of possible queries on the database is presented in section III..

II. Architecture of document analysis using IRS techniques

As illustrated in *Figure 1.*, documents to be analyzed can be grouped into categories, such as bibliographic references, books, catalogues, etc. Some of these categories are well structured while others are not. Knowledge on well structured documents are relatively easy to obtain. For example, the logical and physical structures of bibliographic references are generally normalized. Their logical structures can be obtained from their BIB_TE_X representation and their physical structures can be obtained from the printing style used for generating the reference. The links (a1) (ai) (an) symbolize the process of extraction of the knowledge on the different categories of documents.

We have adopted the approach that consists in transforming the logical and physical structures into their SGML equivalents. This format corresponds to our central format for information representation. We have chosen SGML format for two main reasons. The first reason is that SGML format allows an easy representation of concepts that are related hierarchically or by simple associations. Since document structures are based mainly on the concept of hierarchy or simple association, the SGML format is very suitable for their representation. SGML format can be suitably applied to well structured document as well as those that are not.

The second reason for choosing SGML format is that it can be considered as a grammar which can be easily used for programming. It can be viewed as a syntactic representation of a document and semantic interpretations can be attached to the tags. Furthermore, since there is no limit to the tags that can be used in SGML representation, any type of document can be represented using SGML. Therefore, the adoption of this format allows us to represent well structured documents as well as those that are not.

From the SGML representation of our knowledge on the document structures, we generate an Object Oriented Database (OOD). Here again, the use of SGML format for information representation facilitates the generation of the OOD. The main technique employed is to attach a set of instructions to each SGML tag. We use OOD for managing our knowledge on document structures for the reasons inherent from object programming. For example, OOD allows the implementation of the concept of class from which object instances can be created, inheritance, polymorphism, considering each object in domain of application as distinct objects and the possibility of associating particular methods to the instances obtained from the class. The generation of the OOD is illustrated in the figure by the links (c1) and (d1). OOD1 is the corresponding object oriented database for the logical and physical structures of the classes of documents.

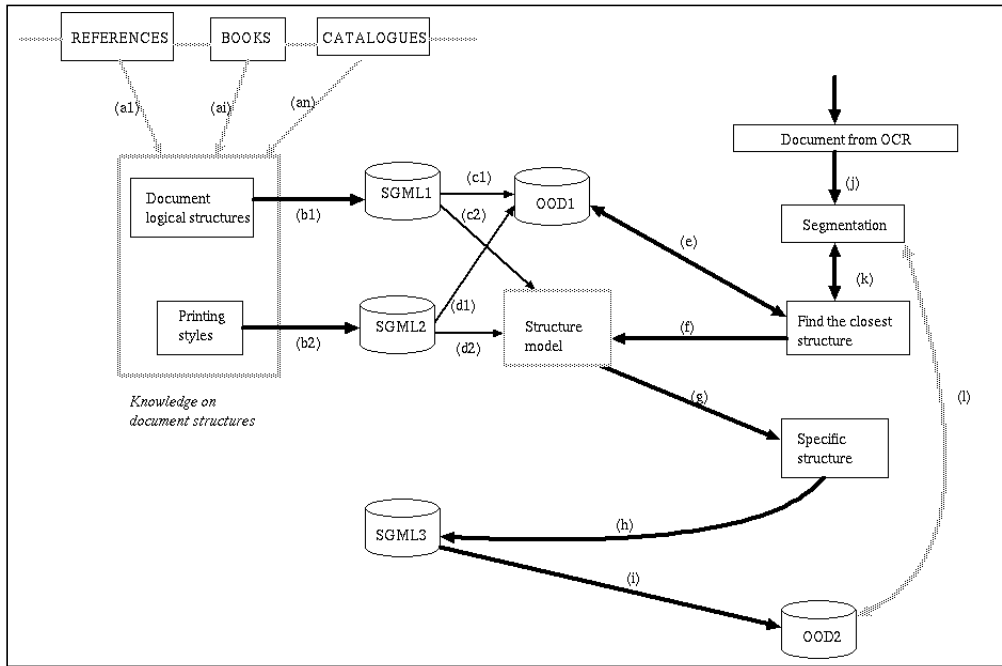


Figure 1. Architecture of IRS in document analysis

As illustrated in *Figure 1.*, we start our presentation of the process of document analysis from the stage of segmentation. The result of segmentation consists in a set of document tokens, that is the basic components of the document analyzed. In order to render the analysis as accurate as possible (that is the identification of the document contents), there is need to identify the document with a document structure (logical and physical structure). The objective is therefore to **find the closest document structure** to the document analyzed. It is at this stage that we propose the use of IRS techniques to discover the closest document structure. In order words, the OOD is used as knowledge base. Following are some examples of information that can be obtained from the OOD (OOD1 in the figure) :

- What are the printing styles that use the printing style discovered from segmentation ?
- The printing style obtained above is associated to what type o document logical structure ?
- What are the components that can be expected from a particular type of document logical structure ?
- What are the separators that can be expected from a particular type of printing style ?
- etc.

After the interactions in (e), (f) and (k) of *Figure 1.*, a **specific document structure** is obtained. This structure contains both the document's logical structures and the associated tokens. This specific document structure is also represented in SGML format (SGML3 in the figure) since the format provides the means for this type of representation. We use SGML3 to generate a new OOD (OOD3) that represents the document and its elements. This second OOD provides the possibility of exploiting the document contents at a semantic level. For example, instead of simply considering the document tokens a simple characters, they can be considered as semantic representations. This extention illustrates how document analysis can contribute to document indexing in IRS.

It should be noted that OOD2 can also be used during the process of discovering the closest structure from the seg-

mentations. For example, the OOD can be queried to discover in what type of document that a particular token is most frequently used. This can guide the discovery of the closest structure.

III. Information analysis using an IRS

The OOD query method that we developed is what we call *classification* with *constraints*. The result of any type of query provides the distribution of values assigned to the document attributes. We present below what classification and constraints are.

Classification allows a user to specify the attributes to use for analysis. Attributes can be combined in three ways :

- *Only one attribute* : Let L be a set of all assigned values to the attribute A in the database. Specifying only one attribute, say A , for analysis, we suppose that the user wants to obtain the distribution of each element of L over the database objects. In other word, the user wants to obtain how many times each element of L is assigned to the attribute A . For example, if the user specifies *Author* as the only the attribute for analysis, we provide the list of all authors in the database and how times they are assigned as authors.
- *Two same attributes* : Let L be a set of all assigned values to the attribute A in the database. If two attributes, say X and Y , are specified for the analysis and that $X = Y$, then we suppose that the user wants to know the distribution of the co-occurrence of any two values of L for attribute X (or Y). For example, if the user specifies $X = Author$ and $Y = Author$ and L is a set of all the authors of the database, then we provide all the co-occurrence of any two authors that have written together and how many times they exist as co-authors.
- *Two different attributes* : Let L be a set of all assigned values to the attribute A and R a set of all values assigned to the attribute B . If the user specifies A and B (A different from B), then we provide all the existing co-occurrences of the elements of L and those of R in the objects of the database. For example, let $A = Author$ and $B = Keyword$. In this case we provide the frequencies at which each keyword in the database is used by each author. These three types of classification can be combined with constraints that indicate the conditions that should be satisfied by the objects to be produced as results.

Constraints can be represented as

$C_1 opb_1 C_2 opb_2 \dots C_n$ where

$C_i = \{attribute, opc, value\}$ where

attribute is one of the document attributes,

opc is one of the comparison operators [$=, <=, >=, *, etc.$]

value the attributes's value

opb_i is a boolean operator [*and, or, except, etc*]

The C_i s correspond to what is called criteria. In IRS, the criteria can be combined using boolean operators like *or, and, except, etc.*

The algorithm for processing a query is based on the use of symbolic indexed arrays and the facility of obtaining an object's attribute values in object representation. Supposing that the user specifies (*attribute1 = Author*) and (*attribute2 = Author*) as his classification requirement the algorithm for processing this request is as follows :

```

For each o in the list of objects {
  put the authors of object o in A (comment : facilitated by object programming)
  While there are more than one authors in A {
    put the first author in f
    put the remaining authors in A
    For each g in the list of authors in A {
      If the element (f,g) exists in the indexed array then {
        append the object o to the element
      } else {
        create an element (f,g) in an indexed array
      }
    }
  }
}

```

Supposing that the user specifies (*attribute1 = Author*) and (*attribute2 = Keywords*) as his classification requirement the algorithm for processing this request is as follows :

```

For each o in the list of objects {
  put the authors of object o in A
  put the keywords of object o in B
  Fore each f in the list of authors A {
    For each g in the list of keywords in B {
      If the element (f,g) exists in the indexed array then {
        append the object o to the element
      } else {
        create an element (f,g) in an indexed array
      }
    }
  }
}

```

The final result of the processing is presented in form of a list of the co-occurrences of the values in the two specified attributes. The list is presented in decreasing order of the number of objects assigned to each co-occurrence. The results of two queries are presented in *Figure 2.*

The inverse lists are used for processing requests with only one attribute and the constraints. For example, if only one attribute is specified, we use the inverse list to obtain all the values of the attributes, the objects in which they are used and of course the number of objects associated. The object programming language we use provide symbolic indexed arrays which facilitates the management of inverse lists.

(a) request -a1 keyword -a2 keyword	(b) request -a1 author -a2 keyword
{ 92 <intelligence_artificielle><systeme_expert>}	{ 13 <Nye,_Adrian><x_window>}
{ 67 <apprentissage><intelligence_artificielle>}	{ 12 <Rozenberg,_Grzegorz><reseau_petri>}
{ 65 <intelligence_artificielle><representation_connaissance>}	{ 10 <O'Reilly,_Tim><x_window>}

Figure 2. Example of co-occurrences for pairs of two Keywords (a) and pairs of Authors and Keywords (b)

IV. Conclusion

We showed in this paper how some IRS techniques can be of help to document analysis and recognition for the extraction of knowledge and its automatic representation in form of database. Two categories of knowledge on documents are represented. The first category of knowledge concerns the structural organization of the class of documents to analyze. The knowledge in this category is occasionally available in a normalized form, such as BIB_TE_X for bibliographic references or SGML documents for books. The second category of knowledge concerns the physical structure of documents. This knowledge was validated in different recognition systems applied with a great success on bibliographic reference structure identification.

V. References

- [1] F. Parmentier, Logical Structure Recognition of Scientific Bibliographic References, Fourth International Conference on Document Analysis and Recognition, Vol. 2, p. 1072-1086, ULM, Germany, 1997.
- [2] A. Belaïd, Retrospective Document Conversion: Application to the Library Domain, International Journal on Document Analysis and Recognition, IJDAR, Vol. 1, N. 3, p. 125-146, 1998.
- [3] A. David, Modélisation de l'utilisateur et recherche coopérative dans un systèmes de recherche d'informations, ISKO '97 (International Society for Knowledge Organisation), Lille, France, 1997
- [4] A. Belaïd, J. C. Anigbogu and Y. Chenevoy, Qualitative Analysis of Low-Level Logical Structure, In Electronic Publishing, Document Manipulation and Typography, Eds. Ch. Hüser, W. Möhr and V. Quint, Vol. 6, Issue 4, 1994.
- [5] F. Parmentier, Spécification d'une architecture émergente fondée sur le raisonnement par analogie : Application aux références bibliographiques. PhD Thesis, University of Nancy HP, France, June 1997.