Logical Structure Recognition of Scientific Bibliographic References

F. Parmentier and A. Belaïd
CRIN-CNRS, Campus Scientifique, B.P. 239
F-54506 Vandœuvre-lès-Nancy Cedex, France
Phone : 333 83 59 20 84, Fax : 333 83 41 30 79
e-mail : {parmenti,abelaid}@loria.fr

Abstract

In this paper, we are presenting an approach for the logical structure recognition of bibliographic references. The objective is to produce, for each reference (given in a display format, as postscript), structured data containing the hierarchy of fields recognized. As a result of variation among bibliographic references (in order and typographic format of fields, or writing style of the author, for example), we need a robust and tolerant system architecture. Thus, recognition is performed by a concept-oriented system that uses a model automatically built from a reference database. This model represents the reference fields and includes statistics on the occurrence of their terms. Recognition is achieved by a step-by-step activation of the more pertinent concepts. Each activated concept causes the execution of an appropriate searching agent. This architecture is robust and non-deterministic, allowing a solution even in difficult cases.

1 Introduction

The constitution of a digital library requires the continual update of databases either by importing references from other existing databases, or by recognizing paper sources (old catalogues, journal summaries, reference part of scientific publications, etc.). In both cases, one needs to understand the reference structure in order to convert it into the local format. Usually, a logical structure is searched and is represented by a standard such as MARC, UNIMARC, or more generally SGML.

In this study, our aim is to extract the logical structure of bibliographic references located at the end of scientific papers. These references are given either on paper or within electronic documents in a display format (postscript, PDF, HTML, etc.), without any logical information. The last case is interesting in that it is not necessary to recognize the characters because they are directly embedded in the input file. Although the reference structure is normalized, the variations in the writing make a straightforward recognition of the logical structure difficult: the writing style varies from one author to another; the linguistic structure is very poor (there are no structured sentences and they are often incomplete); there is a mixing of languages and many abbreviations. Furthermore, there are few visual indices (typography style, topology, etc.) in order to guide the identification of field.

In this paper, we will first describe the logical structures inherent to bibliographic references and their representation in the model. Then, the system strategy will be presented and followed by some examples and experimental results.

2 Structure Modeling

The model is determined from a reference database in BIBTEX format and from the printed file of this database, using LATEX. We have used BIBTEX as an example of a word processor which is able to manage the bibliographic style automatically. The model generator is generic and can be adapted to other database formats.

2.1 Reference Structure

The reference structure arises from the database format and also from the observed references.

Our aim is to extract from the given reference a structure similar to the one which would have generated this reference. The hope is to generate such a reference, even if it does not exist in the database. This is a tricky problem for many reasons. First, errors can be found in databases because users are not experts and can sometimes insert semantic mistakes in the fields descriptions. Second, errors can be inserted by users who write the references directly without using an automatic generator. There is a risk of mixing styles and of formatting the reference fields and separators differently. Furthermore, information added by users may not belong to the defined fields.

2.2 Model Construction

We use a concept network in order to represent the generic structure. The concept network is composed of nodes and links. Nodes are divided into two categories: *generic*, corresponding to the structure components, and *specific*, dealing with examples contained in the database. The links represent the conceptual proximity of the nodes

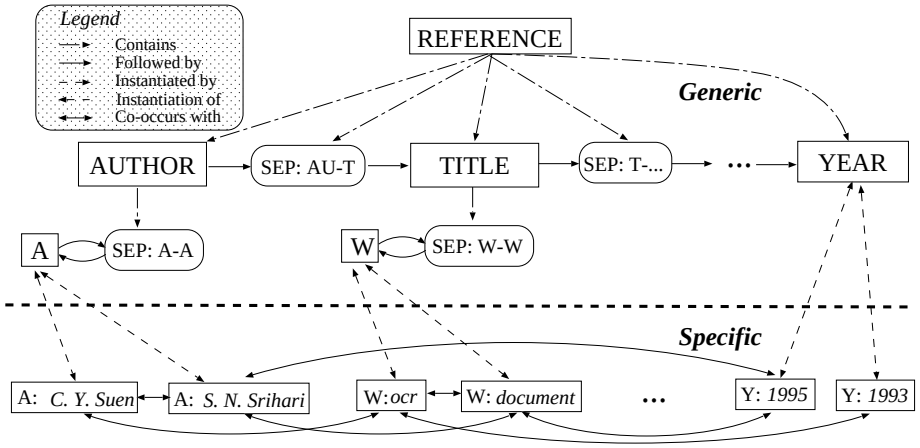

Figure 1: Concept Network.

(e.g. author and co-authors in the AUTHOR field, VOLUME and NUMBER, words in the TITLE, etc.). Figure 1 gives a global view of an example of such a concept network.

Attributes are assigned to each node. We distinguish two kinds of attributes : *fixed*, computed at the net creation, and *dynamic*, evolving during the treatment. The former corresponds to the *name*, the *conceptual importance* and the *activation rate*. The latter gives the *activation value*.

name: indicates the node category (among *field*, *separator*, and *instantiation*) and its significance. (e.g. for *field*, its name, for *separator*, the fields separated and the separator value, and for *instantiation*, the field from which it is an instantiation and the value of the instantiation).

conceptual importance (CI): gives the value of the importance of the node for the resolution of the problem. It is estimated as a function of its occurrence in the database. For an instantiation, the CI depends on the field CI from which it is the instantiation and on the frequency of its appearance in the database. For separators and fields, the CI is proportional to their level in the net hierarchy.

activation rate (AR): indicates the interest in keeping the current node activated (or not) during the system cycles. It is a function of the CI and of the number of links that influence it. This means that the more links there are, the more activated the node is, and thus the AR has to reduce the influence of its links.

activation value (AV): reveals the more important nodes that have to be considered at every step of the treatment. Its value varies at each processing cycle according to the strategy used. We will study this aspect more closely later on.

In order to construct the model, the structural hierarchy (fields and sub-fields) and the terms (field instantiations) are extracted (called logical structure) from a BIBTEX database . For separators, a postscript file is generated and transformed into a more simple SGML format that contains only some text and the typographic style (called physical structure). Then, the physical and the logical structures must match one another in order to find the physical separators (typography and textual separators).

Statistics are then performed on terms, on separators and on fields, by counting their occurrences, and their co-occurrences. The co-occurrences are computed between terms in the same field and in different fields of the same reference. For separators, the co-occurrences are counted only between the separators and the separated fields. Because of the net structure (cf. Figure 1), co-occurrences between fields are not taken into account.

In the end, these statistics are used to weight the links between nodes according to the following formula: $100 \times \frac{\text{NbCoOc}(A,B)}{\text{NbA}}$ where NbCoOc(A, B) is the number of co-occurrences of A and B, and NbA is the number of A occurrences.

3 Analysis Strategy

As described previously, the model used for this application is not based on grammar but more oriented towards the concepts, because users do not respect a rigorous syntax. This leads us to base our analysis approach on the extraction and the validation of terms. Fields are validated by the study of the coherence of each term with its neighbors in the same field and with terms in the other fields.

3.1 System Architecture

The system architecture is drawn from Hofstadter and Mitchell's work [1, 2] on emergent systems. In this system, knowledge is represented by a *slip net*, and evolves dynamically during the treatment in order to adapt the architecture to the given problem. The evolution is managed by a mechanism of propagation of activation pointing

out the more pertinent concepts. These concepts (called "*emerged*") lead to the execution of specific *agents*.

The system BAsCET thus developed is composed of a concept network and a blackboard. The blackboard contains instantiations of concept network nodes, and gives the current solution in terms of objects and links. Each object is described by three attributes: *importance*, *happiness* and *eminence*. *Importance* depends on the CI and the AV of the father node in the concept network. *Happiness* measures the recognition confidence score of the object. At last, *eminence* determines how attractive that object will appear to the agents. It takes into account both the object's *importance* and its *happiness*.

Agents are specific programs (for segmentation, term matching, separator searching, etc.) attached to each concept in the net. When a concept becomes active (its AV becomes high), posts its agents into a *code rack*. At each processing cycle, the system chooses the more urgent agents. Each agent has an *urgency* value deduced from the node AV and adapted according to a *temperature* measurement. The *temperature* measures the degree of perceptual organization in the system (a rough indication of the system satisfaction) and controls the degree of the system determinism [3]. The *temperature* value is equal to the average value of the *eminence* measurements in the blackboard.

3.2 Agents Description

Agents are the procedural knowledge of the system. In an application of logical structure recognition, we defined several types of generic agents. Each one uses the *name* of its launching node to know what is to be considered. When they are executed, they search for all the possibilities that may be present in the blackboard. Thus, once they have been executed, they can be deactivated (they have found either something, either nothing, but in both cases, it is useless to re-execute them in the following cycle). Here are the different types of agents:

field detectors: they are responsible of the detection of the fields. They are of two types:

- **based on content** it looks for the objects that could be inside the field it has to discover to locate it. These objects can be sub-fields, instantiations (of the field, in the specific part of the concept network), or separators included in the field. It is based on the search for on an homogeneous zone containing these objects. When a zone has a high probability for being the field searched, all objects within this zone but that cannot belong to the field and that have not a sufficient internal cohesion are eliminated;
- **based on separators** it looks for separators in which there is the name of the field to discover, and for separators that separate the field containing the field to discover. It mayhave to choose one separator among many candidates. It chooses the best-scored one, in order to discover a field corresponding to the statistical location of this field;

separator detector: it looks in the blackboard for a string similar to the one that it has to find (it is contained in the *name* of the launching node). It uses an edition distance to compute the similarity of two strings. The nearest string is retained only if it has an acceptable similarity. Its *happiness* value is increased by a location score, coming from a statistical separator location base;

instantiation detector: it works like the separator detector, except that it gives different *happiness* value: the highest one is lower than 100%, because, an instantiation can be confirmed by the presence of separators of the field instantiation. It does not use a location score letting it to the separator detector;

stopping agent: it takes the decision to stop the treatment. This decision depends on the *idleness* in the blackboard, and on the *temperature*. The *idleness* is the number of cycles during which there have been no construction or destruction in the blackboard. The higher the *idleness*, the higher the likelihood to stop the process. The lower the *temperature*, the higher the likelihood to stop the treatment.

An agent looks for all possible instantiations of its launching node.

4 Experimental Results and Discussion

Experiments on BAsCET system are conducted using a local bibliographic database containing about 900 references, dealing with the main ICDAR themes. References are corrected by hand in order to insure semantic validity.

4.1 Example of execution

We will discuss in the following paragraphs the execution of the system on the reference given in Figure 2. A processing cycle is constituted of a phase of activation propagation and a phase of agent executions.

Initialization: The first step consists of the initialization of the system. In the blackboard, an instantiation of the root node of the concept network is created (`field:reference`). This instantiation (called an object) contains the reference to be recognized in an SGML format obtained from a postscript file (cf. Figure 2).

At this level, the activation value of the root node is high (100%). Thus, the corresponding object has very high *importance* (I) and *eminence* (E) values (100%) and a very low (0%) *happiness* (H) value because of lack of information at this level. We can also notice that the *temperature* is high as well (100%).

```
<Times-Roman>S. Liebowitz Taylor, R. Fritzon, and
J. A. Pastor.  Extraction of data from preprinted
forms.  </Times-Roman><Times-Italic> Machine Vision and
Applications </Times-Italic> <Times-Roman>, 5:211--222,
1992.</Times-Roman>
```

Figure 2: Input Reference.

First cycle: During the phase of activation propagation, the more important nodes in the network are activated. For this example, the node "SEP:-YEAR-:.</Times-Roman>" is activated, launching the *separator detector* agent. This agent finds the string ".</Times-Roman>" and builds an object in the blackboard (which is an instantiation of the separator node). This object is linked with the field:reference object and the constructor agent activates the YEAR *field detector*.

I	H	E	Content
100	6	100	"<Times-Roman>S. Liebowitz Taylor, R. Fritzon, and J. A. Pastor. Extraction of data from preprinted forms. </Times-Roman><Times-Italic> Machine Vision and Applications </Times-Italic><Times-Roman>, 5:211 --222, 1992.</Times-Roman>" Father node (field:reference), Nb of links = 1
60	100	0	".</Times-Roman>" Father node (sep:year-:.</Times-Roman>) Position: 211-225, Nb of links = 0
Température: 62			

In the propagation phase of the following cycle, more nodes are activated (and some deactivated, according to their activation rate). The following example shows a part of the concept network at the end of this phase.

```
 AV  Node
 85 "sep:-author:<Times-Roman> "
 56 "sep:-author:<Times-Roman>] "
 95 "sep:author-title:. "
 50 "sep:institution-address:, "
 53 "sep:journal-volume:</Times-Italic><Times-
       Roman>, "
100 "sep:month-year: "
 53 "sep:note-:.</Times-Roman>"
 58 "sep:organization-publisher:, "
 66 "sep:school-address:, "
 50 "sep:series-publisher:</Times-Italic><Times-
       Roman>. "
100 "sep:year-:.</Times-Roman>" (sep)
```

The activated nodes launch their agents, and some (a minimum of 60%) are chosen to be executed. These agents build six recognized separators: "SEP:-AUTHOR:-<Times-Roman>", "SEP:JOURNAL-VOLUME:</Times-Italic><Times-Roman>, ", and "SEP:ORGANIZATION-PUBLISHER:, ", and three "SEP:AUTHOR--TITLE:. ", each separator is well recognized except ORGANIZATION-PUBLISHER because it is not located at a usual place in the reference (*happiness value* of 85%). The agents also reactivate other *field detectors*: AUTHOR, JOURNAL, VOLUME, ORGANIZATION, PUBLISHER, TITLE. Other agents are executed, but do not find what they are looking for, they deactivate themselves for the duration of several cycles. The following lines show the blackboard state at the end of this cycle (number 2).

I	H	E	"Content" (Father node) Position: left-right
100	29	100	"<Times-Roman>S. L ... s-Roman>" Father node (field:reference), Nb of links = 8
60	100	0	".</Times-Roman>", 211-225, Nb of links = 0, (sep:year-:.</Times-Roman>)
60	97	0	"<Times-Roman>", 0-12, Nb of links = 0, (sep:-author:<Times-Roman>)
60	100	0	"</Times-Italic><Times-Roman>, ", 165-194, Nb of links = 0, (sep:journal-volume: </Times-Italic><Times-Roman>,)
60	85	1	", ", 205-206, Nb of links = 0, (sep:organization-publisher:,)
60	85	1	". ", 14-15, Nb of links = 0, (sep:author-title:.)
60	82	1	". ", 35-36, Nb of links = 0, (sep:author-title:.)
60	73	2	". ", 51-52, Nb of links = 0, (sep:author-title:.)
60	73	2	". ", 62-63, Nb of links = 0, (sep:author-title:.)
Température: 17			

In cycle number 3, a *separator detector* destructs the object "SEP:ORGANIZATION-PUBLISHER" because it builds another one that has a higher *happiness* value.

In further cycles, other types of agents are executed: *field detectors*, and *instantiations detectors*, which produce content information for the *field detectors* based on the content of the field to recognize.

At cycle 9, another agent decides to stop the treatment. Its decision is based on the low temperature (1) and on the number of cycles during which no building has been done in the blackboard. Table 1 shows the solution contained in the blackboard (L means number of Links).

```
<REFERENCE> 81%
 <YEAR>1992</YEAR>            93%
 <VOLUME>5</VOLUME>           93%
 <PAGES>211--222</PAGES>      93%
 <JOURNAL>Machine Vision and Applications</JOURNAL>                  80%
 <AUTHOR>S. Liebowitz Taylor and R. Fritzon and J. A. Pastor</AUTHOR> 62%
 <TITLE>Extraction of data from preprinted forms</TITLE>             64%
</REFERENCE>
```

Figure 3: Result in SGML (and certainty values).

Figure 3 contains the result produced by the system (with certainty values, i.e. happiness values of the nodes). Note that not all objects are mentioned, as they are not all interesting (e.g. separators). Notice that even if the global certainty value (81%) is not very high,the reference is fully well recognized.

4.2 Discussion

An advantage of system architecture of this type is its genericity: it can deal with various problems (it has already been tested on the academic Traveler Salesman Problem). Also, it can deal with many database formats, thanks to the automatic building of the model and to the agents' genericity. In the previous prototype [3], agents are all specific: there is an agent for each node (e.g. the AUTHOR field detector uses a specific syntax which allows location of this field). In the current version, agents are planned to adapt to any format.

Due to the mechanism of activation propagation in the concept network, the system works by association of ideas. In effect, it looks for terms that are often associated with

I	H	E	L	"Content" (Father node) Position Left-right
100	81	2	13	"<Time ... man>" (field:reference)
60	100	0	0	".</Times-Roman>" 211-225 (sep:year-:.</Times-Roman>)
60	97	0	0	"<Times-Roman>" 0-12 (sep:-author:<Times-Roman>)
60	100	0	0	"</Times-Italic><Times-Roman>, " (sep:journal-volume:</Times-Italic> <Times-Roman>,) 165-194
21	100	0	0	". </Times-Roman><Times-Italic>" (sep:title-journal:. </Times-Roman> <Times-Italic>) 104-133
91	93	1	1	"1992" (field:year) 207-210
19	93	0	1	"5" (field:volume) 195-195
37	93	0	1	"211--222" (field:pages) 197-204
16	80	0	7	"Machine Vision and Applications" (field:journal) 134-164
60	100	0	0	", " (sep:publisher-year:,) 205-206
60	100	0	0	" " (sep:jw-jw:) 7-7
60	100	0	0	" " (sep:jw-jw:) 14-14
60	100	0	0	" " (sep:jw-jw:) 18-18
81	75	2	0	"J. A. Pastor" (field:a) 37-48
92	62	6	3	"S. Liebowitz Taylor, R. Fritzon, and J. A. Pastor" (field:author) 13-61
...	...	...	...	...
100	75	3	0	"S. Liebowitz Taylor" (a:S. Liebowitz Taylor) 0-18
82	75	2	1	"Extraction" (field:word) 0-9
84	75	2	0	"Extraction" (word:Extraction) 0-9
82	75	2	1	"from" (field:word) 19-22
84	75	2	0	"from" (word:from) 0-3
82	74	2	1	"preprinted" (field:word) 24-33
84	74	2	0	"preprinted" (word:Preprinted) 0-9
100	93	1	0	"211--222" (pages:211--222) 0-7
82	75	2	1	"of" (field:word) 11-12
100	75	3	0	"of" (word:of) 0-1
9	75	0	1	"Machine" (field:jw) 0-6
100	75	3	0	"Machine" (jw:Machine) 0-6
100	93	1	0	"Vision" (jw:Vision) 0-5
82	75	2	1	"R. Fritzon" (field:a) 21-30
100	75	3	0	"R. Fritzon" (a:R. Fritzon) 0-9
Temperature: 1				

Table 1: State of the blackboard at cycle 9.

those which have already been found. For example, when a human reads –or hears– "*good*" in a dialogue, he expects to find words like "*luck*", "*morning*", or "*bye*", but no adjective, nor other verbs... It is a kind of context-reasoning, which helps to find a solution more quickly than with a syntactic one (in which all possible solutions, many of which are unlikely, are tested).

The tuning of parameters remains difficult, concerning the model construction, the number of agents to execute at each cycle, and the agents. First, we must tune weights of certain links, conceptual importances, and (AR) of the nodes. For example, separators' CI is determined by hand; AR will have a different impact on the node's AV with a greater number of links. Similarly, the rate of agents that are executed at each cycle influences the results: once the more adapted agents have been executed, is it interesting to execute many others? Furthermore, agents have to deactivate themselves (for how many cycles?) and nodes. There are many parameters in this system, thus it is a tricky task to tune it.

The fact is that the solutions given by the system are sub-optimal. The *stop* agents can stop the treatment before all the recognizable parts of the reference are treated. The system generally stops when the solution is satisfactory (but not necessarily complete), or when it cannot find a more accurate one. This avoids to search an unreachable solution, making the system robust.

In the concept network, many nodes represent the same concept. A term can have many declensions: plurality, conjugation; it can also have various abbreviations, synonyms. These abbreviations appear often in conference titles. This leads to plenty of nodes that should be unified in a single one. To perform this unification, we need a linguistic treatment.

5 Summary and Future Work

We have developed a general architecture based on agents' cooperation and on concept emergence. It is written in C++, and can be used for many applications. We used this architecture on the recognition of the logical structure recognition of bibliographic references. We have proposed a way of *automatically* building the model, from existing bibliographic databases and their physical versions. The system is versatile enough to adapt to almost any database format. It has a "*human-like*" way of reasoning that makes it robust and error-tolerant: it does not stop as soon as a problem does not correspond to a grammar, and it does recognize some words even if they are badly written.

We tested BAsCET on a reduced database. Known references were well recognized (95%), unknown references have about 50% happiness, with mistakes, but this is normal, considering that many separators were absent from the database. We are working on the generation of a wider model, and on the adaptation (and perhaps creation) of agents. Future work will consist of a linguistic pre-treatment of the model, in order to make it more dense and efficient. It would be interesting to study the use of the concept network in reverse to find references in a database from a set of terms (AUTHOR, WORDS, YEAR of publication, etc.).

References

[1] D. R. Hofstadter and M. Mitchell. The Copycat Project: A Model of Mental Fluidity and Analogy-Making. In J. Barnden and K. Holyoak, editors, *Advances in Connectionnist and Neural Computation Theory*, volume 2, pages 31–113. Lawrence Erlbaum Associates, 1992.

[2] M. Mitchell. *Analogy-Making as Perception : A Computer Model*. MIT Press, 1993.

[3] F. Parmentier and A. Belaïd. Bibliography References Validation Using Emergent Architecture. In *Third International Conference on Document Analysis and Recognition (ICDAR'95)*, volume II, pages 532–535, Montréal, Canada, Aug. 1995.