# Bibliography References Validation Using Emergent Architecture

F. Parmentier and A. Belaïd

CRIN/CNRS – INRIA Lorraine
Bâtiment LORIA – Campus scientifique BP 239
54506 Vandœuvre-lès-Nancy CEDEX FRANCE
E-mail: {parmenti,abelaid}@loria.fr

**Abstract**

*In this paper, we present an AI approach for the semantic recognition of Bibliography references. The objective is to produce for each reference (given by an* OCR *flow), a structured data containing the list of the different sub-fields recognized and semantically validated. The validation is operated according to a Bibliography reference database, by the exam of principal terms in each reference field. The system uses an emergent architecture containing a Concept Network built from the database. This net represents the principal fields of the references and includes statistics on the occurrence of their terms. Validation is achieved dynamically by activation at each time of the more pertinent concepts. These concepts verify the presence of their terms by the execution of appropriate agents. This architecture is robust and non-deterministic allowing to find a solution in spite of* OCR *errors.*

**Keywords**: Bibliography References – Emergent Architecture – Concept Network – Term Cooccurrence.

## 1 Introduction

For many applications, it is necessary to analyze the document content and to extract the structure relatively to its finest components. This concerns the content architecture of text physical blocks such as *footnotes*, *figure captions* and *Bibliography references*. For these blocks, physical structure (limited to a short line sequence) is less informative than logical one which is able to inform the system about components structure and content. This information, which is fundamental for recognition, deals with typographic style, term linguistic affiliation, contextual information on successive sub-fields such as limits and location. In this case, the structure recognition becomes document understanding. Several document image understanding systems have been proposed for documents such as letter addresses [3], forms [2] where the logical structures is more complete than layout structure.

In this paper, we present a system for the recognition of Bibliography references. The logical structure is composed of a series of fields corresponding to *authors*, *title*, *conference name*, etc. Limits between these fields are not always easy to detect and there is not a rigorous syntax to describe the field contents. We chose to validate limits between fields and their content by looking for the presence and the cooccurrence of their principal terms according to statistics calculated from a reference database. The literature mentions several algorithms to validate terms linked together, such as decision trees, grammars or discrete and stochastic relaxation. Due to the fact that the validation problem is combinatory (NP complete), all these approaches are time consuming. We chose to use BAsCET, a new architecture, inspired from Hofstadter and Mitchell's work [4], designed to build any type of system that needs interaction between several knowledge sources, declarative and procedural, and that has to show an intelligent behavior, versatile in its decisions, aiming to approach the human cognitive faculties.

## 2 System Overview

The validation is done in three steps (*cf. figure 1*). The first step concerns the syntactical analysis of the OCR flow in order to extract the reference fields. This step, already operational (cf. [1]), is based on search of field initials and finals from a reference grammar.

The second step deals with *term extraction* or filtering in each field. According to each field, terms correspond to names in AUTHOR field, keywords in TITLE, specific words in CONFERENCE field, etc.

The last step is the *validation* of the reference fields, using their term co-occurrences. The aim is to validate the orthography of each term, verifying that these terms already appeared in the base. The more they appeared, the more they are validated. But their relationships in the base are also important: the more often two authors write together, the more probable their names are to be correct if they both appear in the same AUTHORS field.

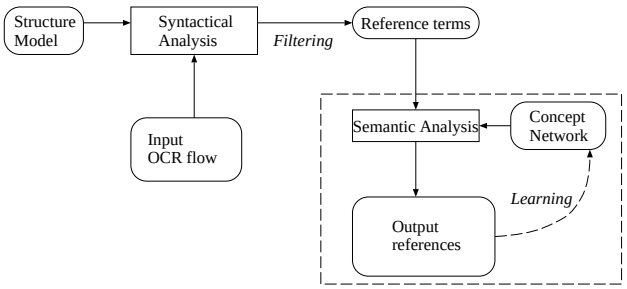


Figure 1: System Overview

## 3 Concept Network

As shown in figure 2, the Concept Network is separated into two parts: the *generic* part which corresponds to the syntactical aspect of the model and considered as a *long term memory*, and the *specific* part which contains the domain information and can be considered as a *mean term memory*.

### 3.1 Generic part

The generic part of the Concept Network represents the system model of the domain (the references). Its is a hierarchy of nodes corresponding to classes and sub-classes. The root node represents the reference concept, while the intermediate nodes represent different fields such as AUTHORS, TITLE and CONFERENCE. Leaves correspond to elementary parts of different fields, such as NAME and FIRST-NAME for the field AUTHOR. There are two kinds of links between nodes: *composed-of* and *co-exists with*. The later expresses the fact that nodes belong to the same conceptual level in the hierarchy.

### 3.2 Specific part

The specific part, obtained by "compilation" of the reference database, allows to point up the most relevant terms in all the references. It corresponds to

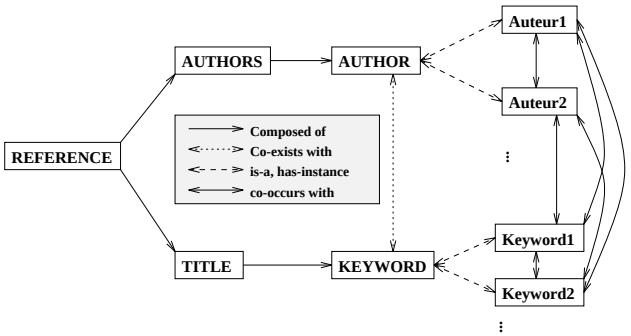


Figure 2: Concept Network

nodes given by instantiation of leaves of the GENERIC part, such as Author$_i$ for the sub-class *Author*. Links between generic part and specific part are of type: *is-a*, while links within specific part are of type: *co-occurs*. This link expresses the fact that the two terms (nodes) appeared in the same reference.

### 3.3 Emergence Principle

The emergence principle is based on the activation of nodes. A node emerges when it is activated. Each node has an activation value depending on the influence of its neighbors, updated at each cycle. The calculus of the influence is based on the neighbor proximity and activation. The proximity between nodes depends on the type of their link and is often expressed as a ratio between occurrence values. Here are the formulae giving the proximities of *generic nodes*:

- *composed-of*: the proximity value is a constant given by the designer,
- *co-exists with*: (ex: A coexists with B) :

$$C1 + (100 - C1)\frac{\text{NbA}}{\text{NbB} \times \text{NbT}}$$

where NbA (respectively NbB, NbT) is the number of A's (respectively B's, T's) instances in the base, C1 is a constant set by the designer, and T is a node composed of A and B.

The other types depend on the reference database.

## 4 Construction of the Specific Part

The system counts in the database, for each node of the generic part, the number of its occurrences, the

number of its co-occurrences with the other nodes, and deduces the proximities with nodes appearing in the same reference.

We give in the following the formulae for the calculus of links used at this level

- *is-a* (ex: **A is-a B**):

$$C2 + (100 - C2) \times \frac{\text{NbA}}{\text{MaxOccB}}$$

  where NbA is the number of A occurrences, MaxOccB the maximum of occurrences of all the instances of B, and C2 is a constant (given by the designer), this is the same for the second formula.

- *has-instance* (ex: **B has-instance A**), this link is the inverse of *is-a*:

$$(100 - C2) + C2 \times \frac{\text{NbA}}{\text{MaxOccB}}$$

- *co-occurs with* (ex: **A co-occurs with B**)

$$100 \times \frac{\text{NbCoOc}(A, B)}{\text{NbA}}$$

  where $\text{NbCoOc}(A, B)$ is the number of co-occurrences of A and B.

The proximities between the node AUTHOR and the authors represent the number of occurrences of the authors in the list. The proximities between the authors represent the number of times they have been co-authors in the list.

Let (C,B), (C,B), (A,B), (C) and (B) be the author's list in five references. Figure 3 gives the occurrence number of each author (A: 1, B:4, C:3), the number of their co-occurrences (AB:1, AC:0, BC:2), and the proximity values which calculus detail is given by the grey areas.

## 5 Architecture Functioning

The initialization of the system is the activation of the Concept Network. This is done by creating an object (thus, an instantiation of a node in the Concept Network) in the Blackboard, activating the node of whom it is an instantiation. Then, the agents of the activated node(s) are posted in a queue (called Coderack in the COPYCAT architecture).

Next, N agents are extracted (probabilistically chosen, depending on *temperature* – which measures how much problems are solved – and on agent urgency values) from the Coderack and executed. The number N is a parameter of the system (Mitchell assigned it to 15 in the COPYCAT project). After that, Concept Network is updated, that is to say its activation is propagated between the nodes. Then, activated nodes are allowed to launch agents.

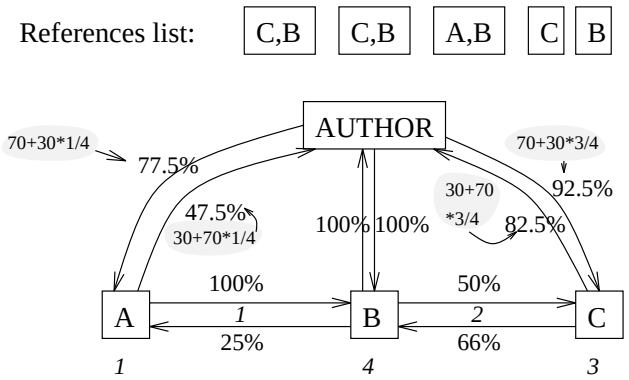


Figure 3: Piece of the specific part of the Concept Network

This re-iterates until a stopping agent is executed and actually stops the process.

The solution is build from the blackboard state, and this state is optionally added to the references base, in order to be later integrated in the particular part of the Concept Network.

### 5.1 Agents

There are currently two types of agents: the first one is the *validating agent*, which decides whether or not there is a term in the Blackboard (instantiation of the node AUTHOR (respectively KEYWORD)) corresponding to the author (respectively keyword) that launched the agent. This decision is taken after a comparison using the edition distance. When 75% or more of the term letters are the same (in the same place), the term is recognized and its confidence score is saved.

Another type of agents is the *stopping agent*. Each node of the particular part in the Concept Network have such an agent. Its urgency value is lower than the other agents, so that it has low probabilities of execution as long as other agents are present in the Coderack. Once executed, it examines the system idleness (when an agent changes something in the Blackboard, the idleness is reset to zero, else the idleness increases at every update of the network), and when it is greater than the permitted idleness, it decides probabilistically (depending on the system's temperature) whether or not it is time to stop. It then shows

the recognized reference (with the confidence scores).

The beginning idleness is a parameter of the system. If an action is done in the Blackboard (descriptor change, object creation, ...), and the idleness is near of the maximum permitted idleness (with a 20% margin), this permitted idleness is increased of 20%. This behavior allows the system to stop early, and it still is likely to execute a last-action agent.

### 5.2 An example

The Concept Network (cf. figure 4) is obtained from a reference database similar to the one shown in figure 3, added with some keywords. At the beginning, all the Concept Network is inactive.

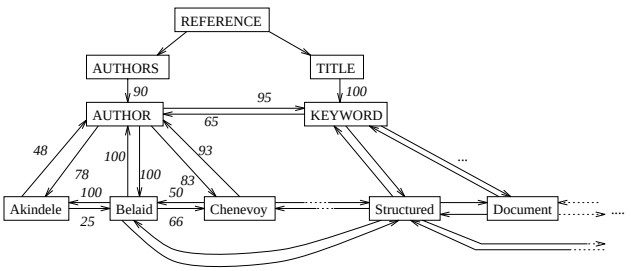


Figure 4: Concept Network of the example

After introducing the text of reference containing two authors, and a title with two keywords, the system is the state shown by the figure 5. All the nodes which were recognized are activated (100%), and the others begin to emerge (in case of failure, all solutions have to be tested). The process took 4 cycles to produce its response: it found and recognized 2 authors, and 2 keywords (this is an optimal case because all the terms were already in the base). 3 decomposition agents and 4 validating agents were executed, which took more time than the 18 stopping agents that only checked whether or not the system should stop. The response was quasi-immediate (the Concept Network was small, but tests with more nodes gave similar speed).

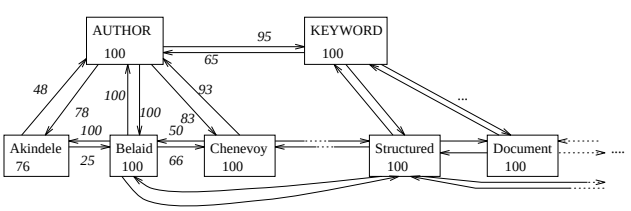


Figure 5: Final state of the Concept Network

## 6 Conclusion and Discussion

We have developed an architecture based on agent cooperation. The principal information is represented by a Concept Network which is a semantic net evolving dynamically during the system execution. The evolution leads the system to propose, at each time, the interesting concept to be analyzed and appropriate agents to execute on this concept. To avoid determinism of classical models, we use a temperature measurement which controls the system decisions and by consequence the system convergence. This architecture has been used on two applications: traveler salesman problem and bibliographic reference validation. Although the system is in its beginning, it works quickly and solutions are satisfactory: they correspond to solutions given by human and speed is acceptable. For the first problem, since each town is given with distances from all the others, the solution is found in only one cycle. But for references, the system is tested only on the author concept. The activation values are correctly propagated allowing to tune the system parameters. The architecture is completely implemented (in C++) on a Sparc Station.

## References

[1] A. Belaïd, Y. Chenevoy, and J. C. Anigbogu. Qualitative Analysis of Low-Level Logical Structures. In *EP'94*, volume 6, pages 435–446, Darmstadt, Germany, Apr. 1994.

[2] Y. Belaïd, A. Belaïd, and E. Turolla. Item searching in forms : Application to french tax forms. In *ICDAR'95*, Montréal, Aug. 1995.

[3] A. Dengel. Document Image Analysis - Expectation-Driven Text Recognition. In *Proceedings of the Workshop on Syntactical and Structural Pattern Recognition*, pages 78–87. International Association for Pattern Recognition, 1990.

[4] 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.