A Model for Computer Analysis and Reading of Text (CARAT): The SATIM Approach

Jean Guy Meunier, Université du Québec à Montréal
Ismail Biskri, Université du Québec à Trois-Rivières.
Dominic Forest. Université du Québec à Montréal
meunier.jean-guy@uqam.ca

Abstract

Computer assisted reading and analysis of the text (CARAT) is a complex technology that connects semiotic text interpretation theories with computer implementation and combinatorial logic. SATIM is a modular computer platform that supports management, itinaries and experimentation of constituent modules (linguistic and mathematical) for expert reading and analysis of text.

KEYWORDS: CARAT technology, Expert reading, Analytic modeling.

Introduction: The Problem

After fifty years, technologies for computer assisted reading and analysis of text (CARAT) have penetrated the various humanities and social sciences disciplines. CARAT technology is found in philosophy (McKinnon, 1968, 1973, 1979; Meunier, 1976, 1997; Lusignan, 1973), psychology and sociology (Barry, 1998; Alexa & Zuell, 1999a, 1999b; Glaser & Strauss, 1967), literature (Brunet, 1986; Hockey, 2000; Fortier, 2002, Bradley & Rockwell, 1992; Sinclair, 2002), textual semiotics (Ryan, 1999; Rastier, 2002), social sciences (Jenny, 1997; Fielding & Lee, 1991; Lebart & Salem, 1994; Mayaffre, 2000), and historical studies (Greenstein, 2002). This technology is different from the AI approach to discourse analysis (Hobbs, 1990; Marcu, 1999, 2000) or automatic reading (Ram & Moore, 1999) where the objective is to have the computer simulate an "understanding" of a text, in some specific application or process (i.e., inference, summary, knowledge extraction, question answering, mail routing, etc.). CARAT technology is also different from information retrieval (Salton et al., 1995a, 1995b) or hypertext technologies (Rada, 1991) where the objective is to find documents for a particular query or to navigate through documents. In CARAT, literary critics, philologists, content analysis specialists, philosophers, theologians, psychologists, historians, and any other

type of professional text readers (i.e., lawyers, journalists, etc.) do not accept automatic text "interpretation" tools under any form whatsoever. Rather, they require sophisticated tools to assist them in their own process of interpretation, whether it is through reading or analysis.

Computer technology offers helpful possibilities to the reading process. Archives of electronic texts are now common (e.g., *Oxford Text Archive*, the *Brown Corpus*, *Perseus*, *Gallica*, *Frantext*, and so on); they are more critically edited and standardized, as can be seen in SMGL, XML, and HTML, and can be explored with "intelligent" tools such as navigators, search engines, and hypertexts. Computer technology has also been an imperative part of the analysis process. There are four distinguishable generations of such technology. The first generation (1950-1970) allowed the capturing of electronic text. A second generation (1970-1980) offered tools for the description and manipulation (extraction, concordance, statistics, and lemmatization) of these electronic texts. A third generation (1980 -1995) started a standardized tagging (SGML, XML, TEI initiative) and linguistic processing (syntactic, semantic, discursive, and rhetorical) of electronic text. A fourth generation (1995-present) introduced sophisticated mathematical models such as classifiers and categorizers on the text. Currently, there exists a proliferation of such tools: from the more hybrids to the very specialized, from the laboratory prototypes to the more industrial applications (e.g., Concord, Word Cruncher, Tact, Cobuilt, Word Smith Tools, Lexa, Cosmas, Tropes, In-text, Alceste, Sphinx, Hyperpro, Atlas, and Nu*DIST). CARAT is associated with a buzz word developed in the commercial field, known as *Text Mining*, for which there is a multitude of tools. Unfortunately, the traditional communities of the humanities and social sciences (with slight exception in terms of critical editing of texts, qualitative content projects, or historical archives) do not always recognize the importance of this technology. Text mining has not been fully integrated into practise. In the majority of cases, reading and analysis is still done in a classical manner albeit helped here and there by computer technologies.

Since 1990, many researchers have tried to understand the reasons for this hesitant attitude towards computer assisted reading and analysis. In 1992, S. M. Hockey noticed that these technologies have not obtained the same success in the humanities and social science as in other disciplines. One reason advanced is that they were never fully adapted to the dynamics of reading and analysis of text, as practiced by other disciplines. One proposed solution has been the tool box approach [*Text Software Ini-*

tiative (Ide, 1992)]. In its original objectives (1992) the TACT system (Bradley & Rockwell, 1992) also had integrative views for flexibility and modularity. However, this project did not realize all of these objectives. During a CETH meeting regarding Text Analysis Software for the Humanities, at Princeton University's Carnegie Center in 1996, S. Hockey reformulated the same questions of flexibility and modularity. Recently, Unsworth (2003), commenting on the Hockey critique, declared: "the same paragraphs could be written today, word for word, and it would still be as true as it was in 1996." Even though computer technologies are accepted more for classical transcription, navigation, and configuration of texts, they encounter many difficulties when applied to CARAT. Many researchers believe in it, yet, they are still unsatisfied by the methodologies and tools available.

There are many reasons for these methodological difficulties. Firstly, the weakness of the ergonomics within these technologies renders learning and usage difficult for many users, particularly in the humanities and social sciences (Unsworth, 2003). Except for a few systems, many technologies have been developed by communities external to CARAT (e.g., linguistics, artificial intelligence, data processing, information retrieval), for specific objectives. Secondly, the limited sets of tools available for text analysis, whereby a researcher is confined to text transcription, text encoding, lexical concordances or collocation, basic linguistics (e.g., lemmatization, tokenization, and morphological taggers), or statistical analysis. Thirdly, these commercial tools are often geared more towards information retrieval rather than real assistance for reading and analysis of text. Finally, the technology is often a closed one; it is proprietary, rigid, non-modular and, as often underlined, not fully adapted to the dynamics of CARAT projects.

Another, more profound, reason for these difficulties is the lack of understanding regarding the effectiveness of this technology. The mainstream community acknowledges that the computer interprets text, rather than acknowledging the computer as an assistant to human interpretation. Furthermore, the linguistic aspect of text often encounters confusion during reading and analysis. Rastier (2001) readily reminds us that text as a whole is formally different from a sentence of a language, and must be approached with tools different from grammar and logic (Marcu, 2000; Mann & Thomson, 1988). Finally, the classical technologies of CARAT, except for those bearing standards and protocols (Sinclair, 2002), have not fully integrated the modeling tools of the software engineering languages

that have developed over the last 15 years (Braumbaugh, 1991; Levesque, 1998), which would allow modularity, reutilisation, exchanges, collaboration and even automatic code generation. If a renewal of this technology is to happen, then one must have a more thorough understanding of reading and analyzing a "text" with the help of a computer. Therefore: a more theoretical and formal foundation of CARAT technology has to be explored.

A text is a complex semiotic object. Even if a text presents itself as a sequence of natural language symbols, it is more than a linguistic object; it is not a well-developed output of a formal grammar. Text contains an argumentation, a narration, a description, a demonstration, a dialog, a discourse--all dimensions that are not in themselves strictly grammatical phenomena, albeit they may present strong regularities that belong to their genre (Rastier, 2001), rhetorical structure (Mann & Thomson 1988; Marcu, 1999), or their logical structure (Hobbs, 1990). One does not and cannot learn texts as one learns a language. Knowing Danish does not designate the ability to create Danish legends.

Access to the content of text is not strictly an algorithmic process (Eco, 1965, 1992); it is not even a computational object (Meunier, 1997). Reading or analysing text is a more complex procedure of heuristics and pattern recognition strategies. All of which are grounded on complex dimensions such as linguistic and mundane semantic structure, inference, pragmatic memory, culture, social interaction, and knowledge repositories. Hence, it is in the nature of a text not to reveal itself in totality. Every text is polyvalent. It varies its meaning based upon reception (Jaus, 1978) and reader (Eco, 1965). Text does not necessarily reveal itself in a first reading, not even in a first analysis (e.g., the Bible or the Koran). The hermeneutic tradition has been repeating this for the last century and contemporary theories of reading indulge in this thesis (Iser, 1985; Fish, 1980; Barthes, 1970). From this perspective, it is evident that reading and analysing a text is not something that a computer can do; reading and analysing texts are strictly human activities. They belong to human communication.

The computer can play a productive role in the reading and analysis process; but only if it is well-situated as an assistant to such cognitive activities. In this perspective, CARAT appears as a set of algorithmic operations assisting the construction of interpretative paths when reading and analysing texts. The various generations of the technology presented above have offered computer design for CARAT. But they were

often *ad hoc*, resolving specific and limited problems, delivering heuristic applications in various fields but lacking in integration. If the technology requires a renewal, it has to be founded upon a more complex and integrative design. Because of the complexity of such a problem, we must explore a more subtle design for CARAT. In this paper, we shall briefly expose the SATIM approach to this problem.

The design proposed here is presented in three levels: the first, analytical, describes the CARAT process in terms of the methodologies used in various social and humanities practices; the second design analyzes this same process in a functional language; and the third design manages CARAT in terms of the procedures that can be realized in a computer architecture. Merging these three designs allows for a more integrated understanding of the CARAT process. The three designs are built upon the principle that CARAT is a cognitive path of an interpretative or semiotic nature, which an expert reader will follow when processing textual data to produce knowledge; the computer is placed in a secondary position, as technological assistant.

The Analytic Modeling

The first type of modeling design is analytical. Its aim is to identify, in a descriptive but semi-formal language, the constituents of a reading and analysis process as practised by the experts. In an expert reading and analysis of text, a particular methodology is applied that has been acquired in various applications. In the modelling of these practices, it is mandatory to find underlying similarities and regularities among the various methodologies practised by CARAT users. These methodologies are often constrained by epistemological or logical principles. For example, a hermeneutic reading (Iser, 1985; Jaus, 1978) of a text will require the document's authentication, historical lexical interpretation, structured decomposition, argument structuring, rhetorical description, production of commentaries, and explication. Qualitative content analysis may also require these operations (Richard & Richard, 1994) but will add a sampling of the document, semantic and categorical annotation, statistical analysis of tendencies, etc. Structural semiological analysis (Rastier, 2001) may require some of the preceding operations but add, for itself, identification of topics and isotopics, definition of interrelation, etc. Rhetorical analysis (Longacre, 1983; Mann & Thomson, 1988; Marcu, 1999; Polanyi, 1988) parses text for rhetorical structures. Discourses analysis (Jenny, 1997;

Lebart & Salem, 1994) centers on lexical fields. There are many methods and practices in the reading and analysis of texts. Each discipline has distilled from its own practices a certain number of processes that have been adapted for computer processing. If computer architecture is to be refined to accommodate various methodologies, an analytical description is required to identify the basic and relevant components. This analytical design identifies three main components: units of information, operations and processing chains.

The Units of Information

The first components of the analytical model are the **units of information**; one has to identify the basic, if not the atomic, components of the processes. Traditional approaches usually take the word as the basic unit; this is a highly laden theoretical choice (Benveniste, 1966). For what is a word? In CARAT, there are many types of basic units of processing That is, units of information may be more or less granular depending on the methodology used. They can be also parts of words, whole words, sentences, or paragraphs. They are, in fact, the input and output of various operations and processing chains.

The Operations

The operations are the tasks that each expert understands as rudimentary to its methodology. Although these tasks often appear simple, they are in fact very complex interpretative operations, composed themselves of many sub-operations; experts accept them as components or steps in the methodology even though they may not be aware of all the cognitive and algorithmic details underlying each operation. As an example, here are some traditional CARAT operations applied on a simple sentence: *The cats are on the mat*.

	Operation	Input Type	Output Type	Example	Explanation
1	Lexical extraction	Document	Word list	[the, cats, are, on the, mat]	Lexicon of the sentence

	Operation	Input Type	Output Type	Example	Explanation
2	Morphological tagging	Word list	Tagged word list	[the$_{(art)}$, … mat$_{(noun)}$]	Part-of-speech tagging
3	Lemmatization	Word list	Word list	[the, cat, to be, on the, mat]	Lemmatized word
4	Word frequency	Word list	Word/ Number list	[[the,2], …, [mat,1]]	The total occurrence of each token
5	Word tagging	Word list	Tagged word list		A XML category is a word in a formal language
6	Sentence tagging	Sentence list	Tagged sentence list		A XML category is a word in a formal language

Each operation has a specific type of input and delivers a specific output; some operate on words, others on phrases, paragraphs, and whole texts. Here a few more of these operations:

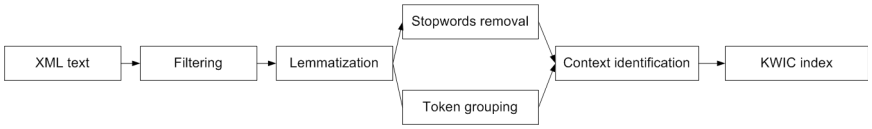
- Choosing a relevant text in a corpus
- Cleaning a document of non relevant information (pre-processing)
- Finding the context of certain words (a concordance)
- Identifying in a sentence the subject and the object of an action, (propositional description)
- Tagging part of a text in XML
- Examining the statistical tendencies of components
- Identifying the distribution of a theme in various documents
- Comparing a particular textual segment of a text to other segments, to expose lexical or stylistic differ-

ences
- Adding a semantic or syntactic description to certain lexical items

The input of each operation can be various linguistic semiotic forms of the original text (e.g., words, n-gram, sentences, paragraphs, chapters, fragments, or documents). Each operation is not a simple task; it may include many sub-operations. Some operations can only be realized manually; others can require computer assistance and there is a myriad of such operations. Any serious analytical model of CARAT requires the identification of these basic operations for each methodology. These models constitute the building blocks of a solid CARAT analysis chain.

Analysis Chains

A CARAT application is not simply a *set* of operations. It is also a *combination* of operations. Each of which can form an *analysis chain* or an itinerary. In technical terms, an analysis chain or itinerary is a combination of operations; each output of one operation can be the input of one or many other operations as in examples 3 and 5, whereby the output of the lemmatization operation is the input of the XML word tagging. By this composition, basic operations can become more complex, so that a particular CARAT methodology becomes a particular combination of operations. For example, a thematic analysis is a complex combination of many sequential and parallel operations on a text. Here is a simple example of such an analysis chain: a concordance chain.



In this example, we have chosen only a few operations. Many other operations are possible depending on the objectives at which they are aimed. And finding good and relevant combinations is often at the core of a research project. Even if there are stable methodologies and good practises (e.g., qualitative, terminological, stylistic, categorization, or discourse analysis), many combinations may have to be tested and vali-

dated before one is accepted. For instance, how does one build such a chain for authorship identification? What is the best thematic navigation course in a text? It is precisely here that granularity, modularity and flexibility in the definitions of the operations are important. Combinations depend on their quality.

Are there any operations and analysis chains that are more primitive than others? Some authors (Hockey, 1990; Unsworth, 2000; Lancashire, 1996) have proposed a list such as discovering, annotating, comparing, referring, and sampling. Others have distinguished various practices such as lexicometric, socio-semantic, association, logico-propositonal descriptions, and documentary (Jenny, 1997). These operations are interesting and heuristic but they are rather complex cognitive operations and not really methodological steps in a reading and analysis of text. In our own vision, we would not venture to define a set of primitive operations. We could only recognize that some operations are more basic or "granular" than others from various methodological points of view. In some projects, lemmatization is a basic and important operation. Nevertheless, lemmatization could also be a complex operation or a constituent of another, more complex, operation.

Identifying the essential operations requires a more systematic exploration of the intuitive and epistemological knowledge of the methodologies effectively used in CARAT. Initially, one must produce a descriptive and deconstructive analysis of various CARAT applications, tools, and experiments; it will then be possible to effectively extract the basic operations and combinations of operations. This is the essence of analytical modelling. Software engineering methodologies can be of great help to this modelling process, such as object oriented modeling (Braumbaugh et al., 1991), ontology design strategies (Gruber, 1993; Jalloul, 2003), conceptual graphs notation (Sowa, 1991), or even various knowledge representation approaches (Levesque, 1984). Clarifying the systematic nature of the operations allows one to more accurately determine which series of operations, especially which sub-operations, can be carried out by the experts and which ones can be given an algorithmic translation. The analytic modelling is the foundation for a better CARAT technology. However, before identifying some of these dominant operations and analysis chains, we must give these concepts of *units of information*, *operations* and *analysis chains* a more formal definition.

The Functional Design

In the analytical model, one aims at explicitly defining the set of operations and analysis chains. In the functional modelling, these operations and analysis chains are now translated into functional terms. This translation allows a more formal definition of an operation in terms of its internal structure and relation with other operations. Finally, this translation allows for a better specification of the third level of design: computer architecture.

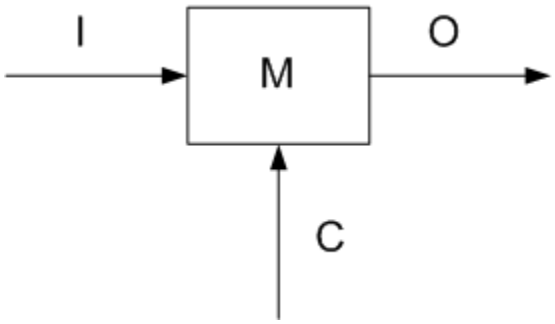
Units as Domains and Values of Functions

In the analytical model the basic components were the *units of information*. These units are both the inputs and outputs of the operation. Hence, they define a domain and a field on which the operations are applied. It is possible to distinguish different types of such units. For instance, *words* are an *object*–type, rather than a *sentence*-type. These types might be words defined as sequences of characters separated by blanks. But, they might also be lemmatized words [e.g., book(s)], morphological tagged words (e.g., book as a name vs. book as a verb), complex words (e.g., bookkeeping), or they may be n-grams as sequences of characters (e.g., books: Boo, ook, oks, …) (Damashek, 1989). In higher levels of analysis, words may be whole clauses (Grimes, 1975; Longacre, 1983), prosodic units (Hirschberg & Litman, 1987), turns of talk (Sacks et al., 1974), sentences (Polanyi, 1988), or intentionally defined discourse segments (Grosz and Sidner, 1986). Distinguishing various types of objects allows the building of operation classes that have the same type of input and output objects. In turn, this allows a higher level of generality for the CARAT functional modelling.

Operations as Functions

In the mathematical sense, an operation has an internal structure. It is defined as functional relation from a set of objects to another (or the same) domain of objects. Given two numerals, such as 4 and 2, the operation of multiplication delivers another numeral, 8, for all numerals in the set of natural numbers. The input objects (units) of a function are applied to its *arguments* (objects to which it is applied) and their output objects are the values (the output objects) they deliver, or, the values on which they

are mapped. An operation is a function that relates objects of some type to objects of another type. However, this relation is realized in a certain way. The function must be effectively applied through a procedure that is distinctively constrained. So a function is described by its input (domain) and its output objects (field), its procedure, and its controls. The domains and fields of a function can be of various types. They can be objects (e.g., numbers or linguistic symbols) or functions themselves. The procedures can be non-computable or computable operations applied to the objects (e.g., move, go to, add, delete, etc.). The controls are the conditions under which the procedure is applied (e.g., the rules and parameters of a procedure). We can represent the structure of a function by the following graph:



Where:

- I is an input: the *domain* of objects on which the function is applied (its arguments).
- M is a procedure: effective *operations* that are applied to the objects.
- C is a control: the *constraints*, rules or parameters of the operation.
- O the output: the *field* or value of the function.

In CARAT, the objects of a function are the units of information. Unfortunately, in many applications, the procedures and controls (rules or parameters) are not always explicit, except in the program itself. For example: a grammatical rule for parsing a sentence is to mix the procedure and the parameters of its application. Here are simple examples of some CARAT functions:

1. A *lexical extraction* function is a procedure whose domain (I) is a set of words (sentences, paragraphs, etc.), and whose operation (M) consists in extracting according to certain rules (C) and the value (O) of the function a list of the word types. For example, a lexical extraction of the following input "The cats are on the mat" would give the following output: [the, cats, are, on, the, mat].

2. A *tagging function* is a procedure that takes two sets: a set of linguistic signs (S1) (e.g., "Peter") and a set of tags (S2), and according to certain rules, delivers C, a doublet [(Peter), (Proper Name)].

3. A *lexical counting* function is a procedure that takes two sets: a set of words in a text (S1) and a set of numerals (S2) and according to a cardinality count, delivers a doublet that gives each type of word the number of its occurrence. The tagging operations then become a tagging function. For example, a lexical count of the following input "The cats are on the mat" would give the following output: [[the,2], [cats,1], [are,1], [on,1], [the,1], [mat,1]].

The translation of operations into functions is only a change of name. It allows rendering more explicitly the internal structure of the operations identified in the first level design, including the procedure, the control, the results, and the domain wherein the operations are applied. From then on, it becomes possible to compare the functions among themselves, either from their input, their output, or their procedures and controls. Some functions are viewed as identical because they are applied to the same type of objects, others because they present a same type of procedure or control. A *lexical counting* operation becomes similar to a *lemma counting* operation if one accepts that a word is a similar type of object as a lemma. In this sense, many operations are similar because they operate based on the type of an object. But the unique rules or control parameters allow for distinction. For instance, a syntactic tagging of a word is an operation similar to a semantic tagging; both take on input words and add a particular tag to it. However, difference lies in the constraints put upon

the operation, which affects interpretation.

Analysis Chains as Composition of Functions

The third components are the *analysis chains* or *itineraries* in a possible set of analysis. Translating operations into functional languages allows for a formal definition of the combination of functions, giving rules for compositionality and development. Indeed, each function can take the output of another function as input. This allows for the building of complex functions. These functions can then be combined with other functions: by composition of functions.

Because each function is in itself an autonomous object and is categorized, it becomes the input for second order functions. This allows a categorical structure of compositions, as found in the categorical grammar (Shaumyan, 2002; Desclés, 1990; Fielding et al., 1991; Biskri & Desclés, 1997), algebra (Montague, 1972), or combinatorial logic (Church, 1941; Curry & Feys, 1958). The SATIM approach (Biskri & Meunier, 2002) explores the possibilities of formalizing the composition of functions of an analysis chain in terms of operators, as defined in combinatory logic. This allows a perfect structural control of the chains. The analysis chain becomes a syntactically well-formed chain of complex functions.

The concept of combinatory logic was introduced by Schönfinkel (1924); it was further discussed by Curry and Feys in 1958. Combinatory logic was developed to analyze paradoxes and concepts of substitution; it also relates to the lambda-calculus of Church (1941). These two systems are used currently by data processing specialists to analyze the semantic properties of the high-level programming languages. The combinatory logic uses abstract operators called combinators. Combinators build more complex operators from more elementary operators. The action of a combinator on its argument is defined by a specific rule called Beta-reduction

(according to the terminology of Curry). This rule establishes a relation (independent of the meaning of the arguments) between an expression with combinators and an expression without combinators equivalent to the first one.

Here are few examples of beta-reduction rules of some combinators:

Combinator **B** of Composition

$$\frac{\mathbf{B}\ f\ g\ x}{\mathbf{B}\ f\ (g\ x)}$$

Combinator **S**

$$\frac{\mathbf{S}\ f\ g\ x}{\mathbf{S}\ f\ x\ (g\ x)}$$

Combinator **Φ** of distribution

$$\frac{\mathbf{\Phi}\ f\ g\ h\ x}{\mathbf{\Phi}\ f\ (g\ x)\ (h\ x)}$$

Furthermore, complex combinators are built upon elementary combinators (**B B S**, for example). The action of these combinators is determined by the enchained application of the elementary combinators starting from the elementary one on the left.

1	**B B S** x y z t	
2	**B** (**S** x) y z t	**B**
3	(**S** x) (y z) t	**B**
4	x t ((y z) t)	**S**

In practise, combinators are used in a typed functional (applicative) system, assigned to schemes of functional types. For instance, the combinators presented above have the following types (α, β, γ, δ are only variables, they are not basic types). The scheme of the functional type of the combinator **B** is: F(Fαγ)(F(Fβα)Fβγ)). The scheme of the functional type of the combinator **S** is: F(Fα(Fβγ))(F(Fαβ)(Fαγ)). The scheme of the functional type of the combinatory **Φ** is: F(FαFβγ)(F(Fδα)(F(Fδβ)(Fδγ))).

Schemes of types make it possible to find the *functional* type of a particular data processing sequence containing one or more combinators, given the function types that form this data processing sequence. With this intention, we must use the type reduction rule of the Universal Applicative Grammar (Shaumyan, 1987): *If we have a function of type Fxy followed by an operand of type x then we obtain a result of type y*

Example:

In the diagram above, we have two functions in a serial data processing sequence. O1 is the output resulting from the application of M1 to the input I1. O1 (O1 = I2) is also the input of the program M2 because of the chaining of M1 and M2. While applying to O2, M2 produces the output O2.

Formally:
O2 = (M2 I2)
However
I2 = O1
Then
O2 = (M2 O1)
However
O1 = (M1 I1)
Then
O2 = (M2 (M1 I1))

According to the beta-reduction of **B**, [M2 (M1 I1)] is equivalent to **B** M2 M1 I1. Then O2 is the output of the complex function application (**B** M2 M1) to the input I1. Therefore, in a serial chaining of two functions, we can build systematically complex functions by the combinatory operator **B**, out of more elementary or granular functions. On the other

hand, the new complex function **B** M2 M1 becomes an elementary function for another chain. In this case, a patrimony of existing functions can be recuperated, hence saving time and energy.

The whole of the system is founded on a calculation of the functional types. Each function has a type that determines the way in which it will be combined with other functions in a data processing sequence. This type, as previously presented, determines the nature of what a function accepts as input and what it produces as output. As an example:

Let us take again the chain **B** M2 M1.

Let us suppose that the types of M1 and M2 are respectively **Fab** and **Fbc**. M1 is regarded as a function that takes input from an argument of type **a** to produce a result (output) of type **b**. M2 is regarded as a function which takes input from an argument of type **b** to produce a result (output) of type **c**.

We then can calculate the type of the whole chain **B** M2 M1 by the following procedure:

B	M2	M1
F(Fαγ)(F(Fβα)(Fβγ))	Fbc	Fab
F(Fβb)(Fβc)	{α = b , γ = c}	
Fac	{β = a}	

The first step types are assigned to the functions M2 and M1. The type schema is also assigned to the combinatory **B**. In the second step, we apply the type reduction rule to types of **B** and M2; we then apply the unification operation. Basic type **b** is substituted to the variable α, whereas **c** is substituted to γ. A complex function (**B** M2) is then formed with the scheme of type F(Fβb)(Fβc). In the third step, we apply the type reduction rule to the types (**B** M2) and M1. As in the second step, we apply a unification operation. Basic type **a** is substituted to the variable β. A complex function (**B** M2 M1) is then formed with the type Fac. This last stage shows that the data processing sequence formed of the sequence M2 M1 takes an argument of type **a** as input and gives a result of type **c** as an output. We now have a new complex function composed of two functions: M1 and M2; the inputs are of type **a**, delivering a result of type **c** in a syn-

tactic manner. In a serial chaining of two functions, we can systematically build complex functions out of more elementary or granular functions by the combinatory operator **B**. The new complex function is considered an elementary function for another chain. Once again, a patrimony of existing functions can be recuperated, hence saving time and energy.

Some Functions in CARAT

Some of the main functions in CARAT include inscription, classification, categorization, and exploration. These four types of functions are further described as follows:

> Type 1: Inscription is set of transformation functions of documents into admissible data for CARAT processing.
>
> Type 2: Classification is a set of manipulation functions on the electronic text.
>
> Type 3: Categorization is a set of manipulation functions on the electronic text and some exterior information.
>
> Type 4: Configuration is a set of functions that assist a travelling thought result of analysis.

Inscription

The first type of functions relates to a document outside of CARAT so that it can be processed into the CARAT chain. In a CARAT project, it is often surprising to discover that a textual document, in whatever formats it is given (paper or electronic), may require many transformations so that it is admissible to a CARAT manipulation. Normally, a text comes as a document physically carried on paper. But it could also come in an electronic form (e.g., HTML, PDF, etc.). All text reading and analysis requires that these external documents be transformed, into a format accessible for CARAT processing.

Some typical operations of these inscription functions consist of locating and manipulating physical documents (manuscripts, monographs, index cards, etc.), to compose the corpus of textual information that will be submitted to the analysis processes. For instance, if the document comes

as a traditional paper, or as a photographic image, it may require many complex sub-operations such as scanning, OCR processing, editing, cleaning (i.e., non-relevant information removal), correcting, and ASCI translations. Even if the document comes in an electronic format, produced by various text processors or other electronic text manipulators, it will often require complex sub-operations such as filtering, XML tagging, or restructuring. In other words, one needs a set of inscription functions that transforms a textual document into a format that is admissible to CARAT processing.

Classification Functions

Once a document text becomes admissible to some type of CARAT processing, it is manipulated in different manners, all of which can be categorized under one main type: a classifying function. Though some are very simple, and others more complex, in general, a classifier is an abstract machine or function that realizes some type of grouping or sorting of objects. In a set theoretical definition, classification is the projection of a partition on objects, so that they are as homogenous as possible. In functional terms, a classification is an equivalence function applied to the objects. These objects must fall under one identical predicate. Control parameters, or rules, differentiate these classifying functions. This is not apparent at first sight; but a formal analysis of many functions reveals classification type. For instance, the addition operation may not look like a classification, but if translated into abstract algebra, the ADDITION is a functional relation between a set of doublets on numbers into a set of numbers. In other words, the ADDITION operation forms a special type of class, called a ring, with specific properties. CARAT also offers such classification operations. It is not always apparent that many functions are classification. Nevertheless, many functions are grouping, clustering, and listing operations on a set of symbols. Here are some examples of such algorithmic classification functions:

- *Lemmatization*: builds a set of words with some common part, defined by a rule;
- *Lexicon extractor*: puts the token words into their type and produces a list as a set of word types;
- *Concordance*: builds a set of fragments whereby one key word has its center;

- *Word clustering*: builds a set of textual fragments that share some common structure depending on the similarity of criterion (e.g., identical words);
- *Complex words Identifier*: forms sequences of words (as an ordered set) according to a specific rule (e.g., object-oriented-relational-database, or real-estate-agent);
- *Sorting documents*: builds a class of documents by some similarity of measure.

Some of these functions may be computable and hence may receive algorithmic translations. But some may not be computable, either for theoretical reasons or for reasons of complexity. Thus, they have to be done manually. This does not alter their functional status.

Categorization

Categorization functions are strongly related to classification functions and are of paramount importance to CARAT. A categorization function is a classification function that is applied to sets of objects of different types (Meunier et al., 2005). In a classification function, the operation groups the objects into a class. In a categorization function, the function also groups the objects, but also relates them to an element of another type or set of objects. In this sense, a categorization is a special type of classification. In CARAT, we can find myriads of such operations. For example:

- Edetic: XML and SGML are categorization operations between parts of text (i.e., words, fragments, etc.) and a set of categories;
- Syntactical: associates grammatical function to words;
- Semantic: associates semantic descriptions to words;
- Pragmatic: associates users, addresses, etc., to words or sentences;
- Argumentative: associates presuppositions to words or sentences;
- Logical: associates a logical role, including premises, minors, conclusion, etc., to sentences;
- Hypertextual: identifies relationships with other parts

of texts (Rada, 1991; Conklin, 1987);

- Documentary: associates indexical role to words or sentences;
- Interpretative: associates a sociological, psychological, or anthropological category to words or sentences;
- Co-occurrence: associates immediate lexical environment to words or sentences;
- Counting: associates a set cardinality to words or sentences.

Each reader and analyst will have his own categorical framework. A literary expert associates categories that are different from those of a sociologist, a linguist, and a philosopher. The operation of categorical description may be carried out in several ways. In order to understand this operation, one must distinguish it from a certain number of other operations. First of all, categorical description must be distinguished from annotation. Annotation (Nielsen, 1986; Marcu, 1999) consists basically of a direct addition of information to the text, inscribing various types of information on the texts, such as comments, relationship, corrections, references, and labels, in the form of notes. Annotation often produces an independent text. Contrary to annotation, categorization adds to the units of the text, clarifications concerning their functions, in accordance with an interpretative template. Categorization projects a specific structure onto the text, marking it with specific structural information and attaching it to complex labels that identify certain properties.

Secondly, categorization must be distinguished from tagging. Indeed, tagging is one among many sorts of possible categorizations. Its aim is to project on the set of objects, a structure of some sort, that is, to relate the set of objects with another set; tagging names the parts of this structure. For instance, one could structure the following sentence: *John loves Mary* into *((JOHN) (LOVES (MARY)))*. The categorical structure of the sentence is rendered explicit. But with tagging, the structure becomes even more evident. For example: $(\text{John})_{\text{noun}}$ $((\text{Loves})_{\text{Verb}}$ $(\text{Mary})_{\text{noun)SV}})_{\text{S}}$.

Finally, a third distinction must be made between the categorization function and the computing strategies that execute this function. Categorizing is a type of classification. The two preceding types of categorization, structural and simple tagging, are not on the same level as to their computationality. In fact, mathematical theory has demonstrated

that not all forms of classification are computationally definable; there is not always a recursive function that defines them. A categorization that associates socio-political labels (e.g., government, institution, deputy, and elector) with expressions does not belong to grammar; whereas syntactical categories do belong to grammar. Even when classes can be recursively enumerated, there may not be any algorithm to express them. As far as texts are concerned, there are not necessarily recursive functions and algorithms for all categorization operations, or if they do exist, they are not necessarily known, yet. Because of these distinctions, categorization must be seen as a complex operation, which cannot always be carried out by automatons. The reader and analyst must often do it manually or semi-manually, occurrence by occurrence. For example, only one reader can say can that in a particular text the expression *Bonaparte* belongs to the category (CHAUVINISTIC EMPEROR). The choice of categorization strategies and categories then depends on the reader's and analyst's intentions: a terminological analysis is not a stylistic analysis, which in turn, is not a study of argumentation or an indexing aid.

Configuration Functions

The last set of important functions in CARAT pertains to the cognitive relation between a user and the computer application of the preceding functions on the text. All preceding functions may produce interesting and relevant results, but may also be accessible in a format that is more or less cognitively transparent. Indeed, the various CARAT functions may produce strict and rigorous results; but they are so large that these results can not be cognitively processed by an ordinary user (Rosenblaum, 1992; Hearst, 1994).

Configuration is the term applied to a set of functions that translate the results, obtained through configuration, into a representation that is cognitively relevant for the user-interpreter. Often, configuration yields results that are too far-reaching and bears a structure that is inaccessible to the user, especially when the corpus is extremely large. For example, a configuration could yield 75 pages of concordances for a request concerning a text. More often yet, a user would build a structured database using the voluminous information arising from the text, which will allow navigation through particular results, according to particular interests. In this case, one needs functions that render the CARAT processing more helpful, cognitively. We have called these: *configuration functions*. These

functions will often come after a classification or a categorization. In one sense, they are also a categorization function but where another set of constraints are present: the cognitive understanding of the user. Here is the first list of such functions:

- *Listing*: presenting the results in a vertical sequence;
- *Mapping*: presenting the results in a structured graph (e.g., tree, map, etc.) (Herman et al., 1999);
- *Visualisation*: presentation of results re-injected into a type of iconic description (Tiles, 3D space graphs, etc.).

The four broad classes of operations – inscription, classification, categorization, and configuration – are functions that any computer-assisted text reading or analysis technology needs to execute. It should be noted that these operations are functions that are defined here abstractly, independently of their computer instantiation.

The Computer Design: SATIM

The preceding analytic and functional design allows the development of concrete computer architecture. It is here where the units of information and functions conceived in the preceding analytic and analysis chains are transformed into a computer application. Hence, units become *data*, functions become *modules*, and analysis chains become *complete applications* (or, sequences of modules). This is the basis of the SATIM architecture for CARAT. This type of architecture is becoming a trend, especially in software engineering, as it applies to text mining.[1]

This architecture is a computer platform that implements the functional principles presented above. An anterior version of such a platform was ALADIN (Seffah & Meunier, 1995) that offered a polyvalent platform for integrating multiple operations requested by a user; but it was closed and proprietary. The SATIM approach goes farther in this vision. It offers a platform that allows one to compose, test and experiment independent modules, have them communicate, and construct well-formed sequences of modules. More specifically, it offers three levels of construction specialized either for a conceiver, a researcher, or an end user. We can conceptualize the SATIM Platform as presenting three levels of architecture that are called the *workshop*, the *laboratory* and the *application*.

1. The workshop

This first level platform is an incubator and a repository for the various modules (functions) that are to be used when building analysis chains. These modules are autonomous, independent, and can be built in various programming languages, which is a condition for maintenance and updating of the workshop. The SATIM workshop is a special type of tool box. It is a set of modules (functions), rather than a set of tools for working on modules. These tools allow modules to communicate input and output according to a specific research objective or process. SATIM relies on three computer programming approaches: the object-oriented design, the multi-agent design, and most importantly, the combinatorial functional design, presented above. In its actual form, SATIM is an immense relational database managing a variety of existing modules (e.g., lemmatizer, matrix builders, classifiers, statistical packages, etc.), and is apt at accepting future modules. Currently, this workshop is only accessible to expert programmers and its design is part of an ongoing software engineering project.

2. The laboratory

The second level of the SATIM platform offers CARAT tools that allow researchers to explore various analysis chains in a transparent manner, while building out of the precedent tools box and repository. The user does not need to be a computer programmer, though a certain degree of user expertise is required in the field of CARAT applications. Through an ergonomic interface, this user can view modules as desired, functioning within one of the analysis chains. The user is assured that these chains are syntactically well-formed, that is: these chains form a complex algorithm and their meaning depends specifically on the tasks at which they are aimed. Even though the modules can be combined in a well-formed manner, they are not semantically relevant for any particular task. In its actual form, two analysis chains have been built: Numexco and Gramexco. These two chains are classifications chains on texts. One works on words as basic units of information, the other on n-grams. Other chains are being built as the research continues: Indexico for indexing, Ontologico for ontology maintenance, Thematico for thematic analysis, and Categorico for automatic categorization of texts.

3. Applications

The third level of the SATIM architecture is for the end user. If, after many tests and experiments, a particular, successful chain is finally accepted, it can be wrapped up as an autonomous application, transparent to this end user, whereby certain parameters are modifiable (e.g., a particular natural language). A chain may have its own interface. If a particular chain does not fit a specific objective and modules have to be changed, one must go back to the laboratory and experiment on new chains. With SATIM, applications of specific chains have been applied to domains such as philosophy (De Pasquale & Meunier, 2002; Forest & Meunier, 2000) and linguistics (Biskri & Meunier, 2002).

Conclusion

Although these three CARAT models are still schematic and general, we believe they give insight to the processes applied to text, and help systematically build a reliable and acceptable technology that clings to the concrete practices of experts in the humanities and social sciences. This is the aim of the SATIM approach. It is not only a technology, but also a complex modelling process for the software engineering of CARAT. We believe that these three levels of design help to better define specifications for a modular architecture, while progressing computer assisted reading and analysis of text. It is through CARAT that we are able to integrate so many of the emerging technologies modelled according to the same philosophy.

Notes

[1] Systems based on this idea are becoming more and more available (D2K, GATE, LinguaStream, etc.)

Works Cited

Alexa, M. et C. Zuell. 1999a. "Commonalities, difference and limitations of text analysis software: The results of a review." Mannheim : ZUMA arbeitsbericht.

Alexa, M. et C. Zuell. 1999b. "A review of software for text analysis." Mannheim : ZUMA arbeitsbericht.

Barbara J. Grosz and L. Sidner, Candace. (1986). "Attention, Intentions, and the Structure of Discourse". *Computational Linguistics*. 12, p. 175-204.

Barry, C. A. 1998. "Choosing Qualitative Data Analysis Software: Atlas/ti and Nudist Compared." *Sociological Research Online*, vol. 3, no 3.

Barthes, R. and H. d. Balzac (1970). *S/Z*. Paris : Éditions du Seuil.

Benveniste, E. (1966). *Problèmes de linguistique générale*. Paris: Gallimard.

Biskri, I. and Descles, J.P. (1997). "Applicative and Combinatory Categorial Grammar (from syntax to functional semantics)." *Recent Advances in Natural Language Processing (selected Papers of RANLP 95)*. Ruslan M. and Nicolov, N. (editors). John Benjamins Publishing Company, No. 136, pages 71-84.

Biskri, I. et Meunier, J.-G. (2002). "SATIM : Système d'Analyse et de Traitement de l'Information Multidimensionnelle." *Actes du colloque JADT 2002*, St-Malo, France, Mars 2002.

Bradley J. and Rockwell, G. (1992). "Towards new Research Tools in Computer-Assisted Text Analysis." Presented at The Canadian Learned Societies Conference, June 1992.

Braumbaugh, J. et al. (1991). *Object Oriented Modeling and Design*. Prentice Hall.

Brunet, E. 1986. *Méthodes quantitatives et informatiques dans l'étude des textes*. Paris : Champion.

Church, A. (1941). *The Calculi of Lambda Conversion*. Princeton University Press.

Conklin, J. 1987. "Hypertext: an introduction and survey." *IEEE Computer* 20, 9 (Sep. 1987), 17-41.

Curry, B.H. and Feys, R. (1958). *Combinatory Logic*, Vol. I. North-Holland.

Damashek, M. (1995). "Gauging similarity via n-grams: Language independent categorization of text." *Science*, 267(5199), p. 843-848.

De Pasquale, J.-F. and Meunier, J.-G. (2003). "Categorisation Techniques in Computer-Assisted Reading and Analysis of Texts (CARAT) in the Humanities." *Computers and the Humanities*, Volume 37, Issue 1, p. 111 – 118.

Desclés, J. P. (1990), *Langages applicatifs, langues naturelles et cognition*. Paris: Hermès.
Eco, U. (1965). *L'oeuvre ouverte*. Seuil.
Eco, U. (1992). *Les Limites de l'interprétation*. Grasset.
Fielding N. G. and R. M. Lee, (editors) (1991). *Using Computers in Qualitative Research*. Thousand Oaks: Sage.
Fielding, N. G. et R. M. Lee. 1991. *Using computers in qualitative research*. London : Sage.
Fish, S. (1980). *Is There a Text in This Class? The Authority of Interpretative Communities*. Cambridge, Harvard University Press.
Forest, D. et Meunier, J. -G. (2000). "La classification mathématique des textes: un outil d'assistance à la lecture et à l'analyse de textes philosophiques." In Rahman, M. & Chappelier, J. -C. (ed.). *Actes des 5es Journées internationales d'Analyse statistique des Données Textuelles*, 9-11 mars 2000, EPFL, Lausanne, Suisse. Volume 1, pages 325 à 329.
Fortier, P. A. (2002). "Prototype effect vs. rarity effect in literary style." In Louwerse, Max and Willie van Peer (eds.), *Thematics: Interdisciplinary Studies*, p. 397–405.
Glaser, B. G. et A. L. Strauss. 1967. "The Discovery of Grounded Theory." *Stategies for Qualitative Research*. Chicago : Adline.
Greenstein, D. I. (2002) *Historian's Guide to Computing*. Oxford University Press.
Grimes, J. E. (1975). *The thread of discourse*. The Hague, Mouton.
Gruber, T. (1993). "A translation approach to portable ontology specifications." *Knowledge Acquisition*, 5, 2, p. 199-220.
Hearst, M. (1994). *Context and Structure in Automated Full-Text Information Access*, Ph. D. Thesis, UC Berkeley Computer Science Technical Report number UCB/CSD-94/836, April 1994.
Herman, I., Melanon, G., de Ruiter, M. M. and Delest, M. (1999). "Latour - a tree visualisation system." Proc. of Graph Drawing '99 (September 15-19, Stirin Castle, Prague, Czech Republic), SpringerVerlag, 392-399.
Hirschberg, Julia, and Litman, Diane (1987). "Now let's talk about now: Identifying cue phrases intonationally." In Proceedings, 25th Annual Meeting of the Association for Computational Linguistics. Stanford, California, 163-171.
Hobbs, Jerry R. (1990). "Literature and Cognition." Lecture Notes, Number 21, Center for the Study of Language and Information, Stanford, California.
Hockey, S. 2000. *Electronic texts in the humanities*. Oxford: Oxford University Press.
Ide, N. (1992). *Text Software Initiative*. lists.village.virginia.edu/lists_archive/Humanist/v07/0031.html
Iser, W. (1985). *L'acte de lecture. Théorie de l'effet esthétique*. Bruxelles,

Mardaga.
Jalloul, G. (2003) *UML* by examples. Cambrige University Press.
Jauss, H. R. (1978). *Pour une esthétique de la réception*. Gallimard.
Jenny, J. (1997). "Méthodes et pratiques formalisées d'analyse de contenu et de discours dans la recherche sociologique française contemporaine : état des lieux et essai de classification." *Bulletin de méthodologie sociologique*. No. 54, p.64-112.
Lancashire, I. et al. (1996). *Using TACT with Electronic Texts: Text Analysis Computing Tools: Version 2.1 for MS-Dos and PC-Dos*. New York: MLA.
Lebart, L. and Salem, A. (1994). *Statistique textuelle*. Paris : Dunod.
Lévesque, G. (1998). *Analyse de système orienté-objet et génie logiciel, concepts, méthodes et application*. Montréal : Chenelière/McGraw-Hill.
Levesque, H. J. (1984). "Foundations of a functional approach to knowledge representation." *Artificial Intelligence*, 23, no. 2, p.155-212.
Longacre, R. E. (1996). *The grammar of discourse*. New York, Plenum Press.
Lusignan, S. 1973. "Informatique et analyse de commentaires philosophiques médiévaux." *Revue internationale de philosophie*, col. 102, no 1, p. 10-18.
Mann, W. C. and S. A. Thompson. (1988). "Rhetorical structure theory: Toward a functional theory of text organization." *Text*, Vol. 8, No 3, p. 243-281,
Marcu, D. (2000). *The theory and practice of discourse parsing and summarization*. Cambridge, Mass.; London, MIT Press.
Marcu, D. et al. (1999). "Experiments in Constructing a Corpus of Discourse Trees: Problems, Annotation Choices, Issues." The Workshop on Levels of Representation in Discourse, Edinburgh, Scotland, pages 71-78.
Mayaffre, D. (2000). *Le poids des mots. Le discours de gauche et de droite dans l'entre-deux-guerres: Maurice Thorez, Léon Blum, Pierre-Etienne Flandin et André Tardieu (1928-1939)*. Paris : Honoré Champion.
McKinnon, A. 1968. "La philosophie et les ordinateurs." *Dialogue*, vol. 7, no 2, p. 219-237.
McKinnon, A. 1973. "The conquest of fate in Kierkegaard." *CIRPHO*, vol. 1, no 1, p. 47-58.
McKinnon, A. 1979. "Some conceptual ties in Descartes' 'Meditations'." *Dialogue*, vol. 18, p. 166-174.
Meunier,J.G. F. Daoust, S. Rolland, "Sato: A system for Automatic Content Analysis of Text." *Computer and the Humanities*, 1976, vol XX.
Meunier, J.-G. 1997. "La lecture et l'analyse de texte assistées par ordinateur (LATAO) comme système de traitement d'information." *Sciences Cognitives*, no. 22, p. 211-223.
Meunier, J.-G., Forest, D. et Biskri, I. (2005). "Classification and categorization in computer assisted reading and analysis of texts." In Lefebvre, C. et Cohen, H. 2005. *Handbook of categorization in cognitive science*. New York: Elsevier, pp. 955-978.

Montague, R. (1972) *Formal Philosophy/Selected Papers of Richard Montague*. Yale University Press.
Nielsen, J. (1986). "Online Documentation and Reader Annotation." International Conference on Work with Display Units. Stockholm.
Polanyi, L. (1988). "A formal model of the structure of discourse." *Journal of Pragmatics*. No 12, p. 601-638.
Rada, R. (1991). *From Text to Expert Text*. Mc Graw Hill.
Ram, A. and Moorman, K (eds). (1999). *Understanding Language Understanding. Computational Models of Reading*. Cambridge (Mass.): MIT Press.
Rastier, F. et Bouquet, S. (éds). (2002) *Une introduction aux sciences de la culture*. Paris : PUF.
Rastier, F. 2001. *Arts et sciences du texte*. Paris : Presses Universitaires de France.
Richards, T. J. and L. Richards. (1994). "Using computers in qualitative analysis." In Denzin, N. and Lincoln, Y. (editors). *Handbook of Qualitative Research*. Berkeley, CA: Sage, p. 445-462.
Rosenblum, L. and Brown, B. (1992). "Guest Editors' Introduction: Visualisation." *IEEE Computer Graphics and Applications*, 12(4), pp. 18-19.
Ryan M. L. (1999). "Introduction" and "Cyberspace, Virtuality and the Text." Both in *Cyberspace Textuality: Computer Technology and Literary Theory*. Ed. Marie-Laure Ryan. Indiana University Press. 1-29 and 78-107.
Sacks, H., Schegloff, E. A. and Jefferson, G. (1974). "A simplest systematics for the organisation of turn taking for conversation". *Language*, 50, p. 696-735.
Salton, G. and Allan, J. 1995b. "Selective text utilization and text traversal." *International Journal of Human-Computer Studies*, 43:483-497.
Salton, G., Singhal, A., Buckley, C. and Mitra, M. 1995a. "Automatic text decomposition using text segments and text themes." Technical Report TR-95-1555, Department of Computer Science, Cornell University.
Schönfinkel, M,. (1924). "Uber die Bausteine der mathematischen." *Mathematische Annalen*, 92, 305-316.
Seffah, A. and Meunier, J.G. (1995). "ALADIN: Un atelier orienté objet pour l'analyse et la lecture de Textes assistée par ordinateur." International Conferencence On Statistics and Texts. Rome.
Shaumyan, S. (1987). *A Semiotic Theory of Language*. Bloomington Indianapolis: Indiana University Press.
Sinclair, S. (2002). "Humanities Computing Resources: A Unified Gateway and Platform." *COCH/COSH 2002*. University of Toronto, May 26-28, 2002.
Sowa, J. F. (1991). *Principles of Semantic Networks*. San Mateo: Morgan Kaufman.
Unsworth, J. (2000). "Scholarly Primitives: what methods do humanities researchers have in common, and how might our tools reflect this?" Symposium

on Humanities Computing: formal methods, experimental practice. London : King's College.
Unsworth, J. 2003. Delivered as part of "Transforming Disciplines: The Humanities and Computer Science". Washington : University of Virginia, DC. Saturday, January 18, 2003.