Extraction of Recurrent Patterns from Stratified Ordered Trees

Jean-Gabriel Ganascia

Laboratoire d'Informatique de Paris 6 - CNRS
8, rue du Capitaine Scott, 75015 Paris, FRANCE
Jean-Gabriel.Ganascia@lip6.fr

Abstract. This paper proposes a new algorithm for pattern extraction from Stratified Ordered Trees (SOT). It first describes the SOT data structure that renders possible a representation of structured sequential data. Then it shows how it is possible to extract clusters of similar recurrent patterns from any SOT. The similarity on which our clustering algorithm is based is a generalized edit distance, also described in the paper. The algorithms presented have been tested on text mining: the aim was to detect recurrent syntactical motives in texts drawn from classical literature. Hopefully, this algorithm can be applied to many different fields where data are naturally sequential (e.g. financial data, molecular biology, traces of computation, etc.)

1 Introduction

Much work has been done in fields as molecular biology [8], music [7] or text analysis to compare sequences of characters. In the past, an important amount of good results have been obtained on the exact matching problems between strings, areas or binary trees [4], [5]. Other approaches have dealt with approximate pattern matching. Some of them used dynamic programming techniques based on the notion of edit distance [9]; however, those techniques mainly consider flat sequential data, such as strings, without any structure. Some authors have attempted to replace flat sequential data by binary or n-ary ordered or non-ordered trees, but the inherent complexity of pattern extraction algorithms makes them intractable in general cases. It has been proved that some efficient procedures exist [11] under strict conditions, but the imposed restrictions preclude their use for practical machine learning problems. This paper shows that, by restricting the structured input to SOT, i.e. to Stratified Ordered Trees, it is possible to build a new efficient pattern extraction algorithm. This algorithm uses as input a huge SOT containing 100,000 or more nodes, and generates

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 167–178, 2001.

clusters of small similar SOTs that appear to have multiple occurrences in the input SOT. The technique can be used in many different applications, the one presented here being text mining, i.e. the detection of approximate syntactical patterns with multiple occurrences in natural language texts. Of course, there are numerous other applications to machine learning in all domains where data are essentially sequential, as is the case with financial data, molecular biological data, etc.

The main body of the paper consists of four main parts. The first section briefly introduces the notion of SOT; the second recalls the classical definition of edit distance and shows how it has been generalized to be able to deal with SOTs. The third section presents the generation algorithm that builds the similarity graph and the clustering procedure which induces patterns with multiple approximate occurrences. The fourth and final section provides some information about the use of the algorithm on text mining and considers its efficiency in practical terms.

2 Stratified Ordered Trees

2.1 Ordered Trees

According to a classical definition, an ordered tree is a tree where left to right order between siblings is significant. All sequential data can obviously be represented with a depth-1 ordered tree. By adding levels to ordered trees, it is possible to organize data in a way that represents implicit background knowledge. For instance, a text, i.e. a sequence of characters, is a list of sentences, each of which is composed of words and punctuation marks. Therefore, it can be represented using a depth-3 tree that makes this structure explicit.

2.2 Sorts and Stratification

Ordered trees increase representation power and it is possible to detect similar sub-trees, with respect to this data organization. It is also possible to extract general patterns that have multiple approximate occurrences. For instance, any specific recurrent sequence of syntactical groups extracted from a parsing tree may be detected without considering the corresponding words or their categories.

Nevertheless, due to the high number of potential pairs of sub-trees recurrent pattern detection is intractable. To make it manageable, nodes are categorized into sorts in

such a way that two successfully matched nodes must be of the same sort. In other words, a match between two trees is valid if and only if the corresponding nodes in the matching are of the same sort.

In addition, we suppose that there exists a total order on the set of sorts and that, with respect to this order, the sort of the son(s) is identical to, or immediately follows that of the father. This constraint defines the so-called stratification and the resulting structure is a SOT — Stratified Ordered Tree —. More formally, by defining an ordered set, $\mathcal{S}$ (set of sorts), and a function *sort*(x) which associates a sort to each labeled node belonging to $\mathcal{L}$, we can specify a SOT as an ordered tree where each node sort is either equal to its father sort or to its immediate successor, except for the root which has no father. In case of syntactical trees resulting from natural language text parsing, it means that the ordered set of sorts $\mathcal{S}$ may contain five categories {Text, Sentence, Syntagma, Category, Word} such that Text < Sentence < Syntagma < Category < Word. Let us note that Syntagmas may correspond to syntactical groups or propositions, depending on the syntactical parser, and that they can be recursive. For instance, the son of a proposition may be a proposition.

3 Edit Distance and Similarity Measure

3.1 Edit model

Edit distances have been widely used to detect approximate string pattern matching [9] and a general overview of these techniques can be found in [1]. Let us just recall here some of the basic principles.

Definition: An *edition* is an operator that replaces one character or one sub-string of a string, or more generally one node of a tree, by another one. For instance, a substitution is an edition if it transforms a character of a string into another one in the same position. An insertion (respectively deletion) which inserts (respectively deletes) a character in a string is also an edition.

Remark: in the following, we note $\mathcal{A}$ any set of editions, i.e. any set of transformations from a string or a tree, and $\mathcal{A}_0$ the standard set of editions composed of the three basic operations, *substitution*, *deletion* and *insertion*.

Definition: An *edit distance* between two strings or two trees is based on the minimum number of editions that transform one string or one tree into another. For

instance, here is an edit transformation based on the standard set of editions $\mathcal{A}$ from the string "WHICH" to the string "THAT":

Fig. 1. A set of editions that derives THAT from WHICH

It follows from this definition that the edit distance from "WHICH" to "THAT", i.e. $edit_{\mathcal{A}}$(WHICH, THAT), is lower than the cost of this transformation (Cf. fig. 1) which is equal to the sum of all the corresponding edition costs, i.e. cost(substitution(W, T)) + cost(substitution(I, A)) + cost(substitution(C, T) + cost(deletion(H)).

3.2 Edit distance between strings

Let us now consider that strings x and y are given as two tables of length n and m, i.e. as x[1..n] and y[1..m]. Then it is possible to build a matrix $(n+1)\times(m+1)$ called $EDIT_{\mathcal{A}}$ where $EDIT_{\mathcal{A}}(i, j)$ is filled with the value of $edit_{\mathcal{A}}(x[1..i], y[1..j])$ for $i\in[1..n]$ and $j\in[1..m]$, while EDIT(0, j) corresponds to the cost of the insertion of y(j) i.e. *insertion(y(j)),* and EDIT(i, 0) to the cost of the deletion of x(i), i.e. *deletion(x[i]).* A simple formula summarizes the way the matrix elements are computed:

$$edit_{\mathcal{A}}(x[1..i], y[1..j]) = \min\begin{cases} edit_{\mathcal{A}}(x[1..i-1], y[1..j]) + deletion(x(i)) \\ edit_{\mathcal{A}}(x[1..i-1], y[1..j-1]) + substitution(x(i), y(j)) \\ edit_{\mathcal{A}}(x[1..i], y[1..j-1]) + insertion(y(j)) \end{cases} \quad (1)$$

where *deletion*(x(i)), *insertion*(y(j)) and *substitution*(x(i), y(j)) correspond to the cost of respectively the deletion of x(i), the insertion of y(j) and the substitution of x(i) by y(j).

This recursive definition is appropriate for computation since the distance between two chains of size n is obtained from the distance between chains with a size less than n. Since the edit distance increases with the size of the chains, the closest pairs will be computed first. So, given a threshold beyond which pairs of chains are not considered as similar, it will be easy to discard pairs of chains when some of their sub-chains have already been discarded.

3.3 Extension of the edit model to SOTs

The edit model can easily be extended to SOTs (Stratified Ordered Trees). It is just to remark that the *left-hand exploration* (i.e. the pre-order) of a SOT unambiguously represents it as a list of node occurrences. Due to the stratification of SOTs, the node sorts refer directly to their level in trees. This representation is unambiguous since if the subsequent node sort is greater than the current node sort, it is its son ; if it is equal it is its brother ; if it is lower, it is one of its ancestors.

Therefore the comparison of sequences of nodes resulting from the left-hand exploration of two trees is equivalent to the comparison of those trees. Taking this remark into account, the edit distance between two SOTs is equivalent to the edit distance between the sequences of nodes resulting from the left-hand exploration of those SOTs. In a more formal way, by denoting *lhe*(T) the left-hand exploration of SOT T, the edit distance edit(T, T') between two SOTs T and T' can be expressed by $edit_{\mathcal{A}}(lhe(T), lhe(T'))$.

1. The Whole Processing Chain

To summarize, the whole processing chain that transforms a natural language text into a set of frequent patterns is given below (fig. 2)

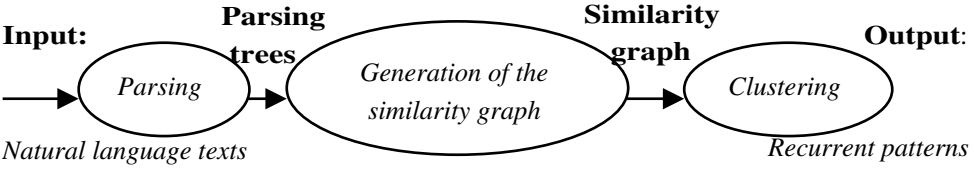

Fig. 2. The whole processing chain

The main input of the system consists of a natural language text that is a sequence of sentences, each being affirmative, interrogative or exclamatory. A natural language analysis performs the first step, through parsing or categorization. It associates labels to words (noun, verb, etc.) and to groups of words (noun group, verb group, etc.). Since the paper is not focused on this natural language analysis, the parser and the categorization process used are not described in detail. We shall just focus on the generality of the approach that has as input any natural language parsing tree with different grammar and different sets of labels. The important point is that the analysis transforms texts into trees or forests, i.e. into sequences of trees, which means that general graphs are excluded. Most of the time, trees are layered, i.e. depending on the

level of the trees, labels belong to distinct classes. For instance, one level corresponds to non-recursive syntagmas, i.e. noun groups or verb groups; the second to word categories, i.e. articles or nouns; the third to attributes like gender or number; the following one to lemma, i.e. canonical forms of verbs or nouns, and the last to words as they appear in sentences. Note that our approach is not restricted to syntactical decomposition in non-recursive groups. The only limitation is that the result of the analysis has to be structured in a Stratified Ordered Tree (SOT)

2. The Similarity Graph

Using the edit distance, a labeled graph called the *similarity graph* is built making the distances between patterns explicit when they do not go beyond a fixed threshold.

4.1.1 Patterns

This similarity graph is of crucial importance; it constitutes the main input of the clustering module and includes all the *patterns* that generalize sub-trees of the input SOT. This implicit generalization is a key-point in the overall algorithm, since it generates all general patterns including non-balanced ordered trees. In the case of natural language parsing trees, generated patterns may look like the following:

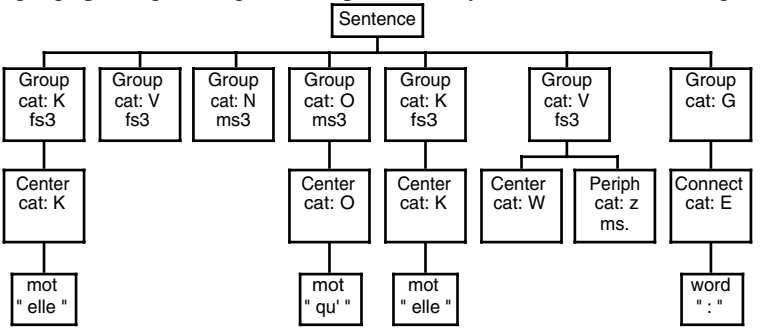

Fig. 3. A non-balanced pattern covering *"Elle exécuta ce qu'elle avait projeté :"*

Note that in Vergne [10] parsing formalism *Group* is a kind of syntagmas that refers to a syntactical group while *Center*, *Periph* and *Connect* are syntactical categories corresponding to the central or peripheral role of the word in the syntactical group and to a connector (here an external connector, i.e. a punctuation mark).

4.1.2 Computing the Similarity Graph

To compute the similarity graph, all pairs of patterns T_1 and T_2 have to be produced and then the value of the edit distance from T_1 to T_2 has to be checked. The upper

bound of the computational complexity of this algorithm is $|S(T)|^2$, $S(T)$ being the set of all sub-trees of T. However, it may be considerably reduced using properties of the edit distance on which the similarity graph relies.

Given any total order $<_e$ on patterns, the symmetry of the edit distance reduces the test to ordered pairs of patterns $\{T_1, T_2\}$ where $T_1 <_e T_2$.

A second important property of the edit distance comes from the restriction to the standard set of editions, i.e. to the three basic operations *insertion*, *deletion* and *substitution*. Let us now suppose that $T_1 \leq_e T_2$, and that $edit(T_1, T_2) \geq \theta$. It is then easy to prove that $edit(T_1, T_3) \geq \theta$ for any T_3 such that $T_2 \leq_e T_3$.

4.2 Pattern extraction

The last step is the pattern extraction. Since the similarity graph records all similarities between patterns, it is natural to extract clusters of similar patterns from this graph. However, the way in which we can build such clusters may differ.

The classical approach is to detect highly connected sub-graphs of the similarity graph [4], [5]. However, for many reasons, this approach appears to be inappropriate to solve our problem. A more satisfactory approach is adopted here. Called the "center star" algorithm, it has been introduced by Gusfield [2] to detect homology on molecular biology data. Then Rolland and Ganascia [7] developed and applied it to extract patterns in music. As this approach has already been published, we shall not describe it here in detail. Let us just define a *star centered* on N as a graph of which all vertices contain node N. In other words, a star centered on N is composed of all nodes P such that the pair {N, P} is a vertex.

To each node N of the similarity graph is associated its centered star that is ranked taking into account two criteria: the number of nodes belonging to it and their similarity to the center. The following formula provides a way to evaluate the centered star associated to each node of the similarity graph and to classify it:

$$star\_value(N) = \sum_{\{N'/\{N,N'\} \in Similarity\_graph\}} similarity(N,N') \qquad (2)$$

Once the similarity graph has been built, this function is easy to compute. However, the evaluation of each arc of the similarity graph, i.e. its similarity, has to be derived from the edit distance that needs to be converted. Given a distance (here the edit distance), many different similarity measures can be introduced such that they obey the classical definition.

Because the edit distance heavily depends on the length of patterns, we have introduced this length in the similarity formula. It means that the error ratio, i.e. the

number of tolerated deletions, insertions and substitutions, depends on the size of the patterns considered. Among the possible formulae the following has been proved to be efficient and useful in practice, even if some others have been tested without substantially modifying the results.

$$s_{\alpha}(i,j) = \frac{1}{1+\alpha \times (\text{edit}(i,j)/\min(\text{length}(i),\text{length}(j)))^4} \quad (3)$$

Remark: α is a positive number acting as a parameter. Its current value is fixed to 0.01 in all our experiments. However, it was experimentally modified from 0.05 to 0.001 without notably changing the obtained results.

Using any similarity measure, the "center star" algorithm first computes all the star evaluations for all nodes of the similarity graph, then the best star is selected and the nodes belonging to it are discarded from the similarity graph before the process iterates.

5 Evaluation

The generated patterns are mathematically specified as recurrent sub-trees, i.e. as sub-trees with multiple occurrences. Edit distance renders possible to extend this definition to approximate occurrences; clustering makes it possible to specify the minimum number of occurrences belonging to patterns. As a consequence, the global evaluation of the overall algorithm does not relate to the nature of the results which are perfectly specified, but to the practical complexity and to the usefulness of the system. This is the reason why it was evaluated on some practical application.

5.1 Application to syntactical pattern extraction

Among many applications of our algorithm, we focus here on the extraction of syntactical patterns resulting from syntactical analysis of natural language texts. The goal of this application is to detect some recurrent syntactical patterns in natural language texts, i.e. patterns with multiple approximate occurrences. There are three reasons for doing this research. The first is to characterize the personal style of authors as we claim that the style of writing is embedded within their choice of syntactical structure and lexicon. The second reason is educational. Our aim is to help school children, students or young writers to evaluate the richness and the diversity of their own writing style, to identify classical mistakes and propose corrections. The third reason is an academic one. The science of language, linguistics, could take

advantage of such a study in order to distinguish different registers of language and to characterize them.

The system was tested on more than 75 18th and 19th centuries short stories and novels written by Madame de Lafayette, Guy de Maupassant, Alphonse Allais, Marcel Schwob, Alphonse Daudet, Eugène Mouton, Hégésippe Moreau and George Sand, among others. The texts were first parsed using the Vergnes-98 analyzer [10] and then the resulting sequence of syntactical trees was transformed into one SOT.

5.2 Efficiency

We studied the empirical complexity by relating execution time in seconds to input size in thousands of words, by reporting it on a log-log scale, and by applying a linear regression algorithm. It clearly appears (see figure 4) that the regression coefficient (i.e. the slope of the line) is equal to 2, which empirically shows that the temporal complexity is quadratic.

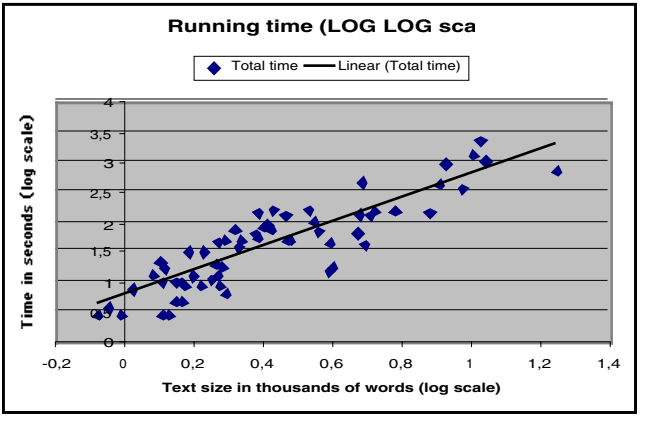

Fig. 4. Empirical Evaluation on 75 Short Stories and Novels (with the same parameters)

This first empirical result is highly satisfactory since the theoretical complexity of our algorithm is at least quadratic with the size of the input text. It comes from the way our algorithm computes the similarity graph, by exploring all the pairs of patterns. Because of the tree structure of texts, the number of sub-trees is linear with the number of sentences, so the global complexity cannot be any lower. To avoid misunderstanding, it should be said that in the case of exact repetition [4] the procedure is clearly more efficient, but not in the case of approximate matching, as it is the case. On the other hand, the center star algorithm that is a greedy algorithm,

appears to be linear with the size of the similarity graph, i.e. quadratic with the number of nodes it contains. Here again, the complexity should not be any lower.
The system has been implemented in C++ and tests are run on a Macintosh G3, with a 300 MHz processor. Extracting patterns from short stories takes a few seconds while, in the case of full novels, it may take one hour or more. This means in practice that it is possible to apply our algorithm to extract patterns that are characteristic of full books, but not to deal with the lifetime work of an author. However, as we shall see in the next subsection, it can already be of great help.

5.3 Examples of extracted patterns

The pattern extraction program is completed by a discrimination procedure. Given two texts, this procedure detects the recurrent patterns covering multiple occurrences of some syntactical structure in the first text without detecting any occurrence of this structure in the second. This discrimination procedure has been employed to detect syntactical structures characteristic of one author, i.e. that distinguish this author from others. The author chosen was Madame de Lafayette, and the two texts, a short story entitled *La comtesse de Tende* and a famous novel, *La princesse de Clèves*. More than 25 short stories from three 19th century authors, Guy de Maupassant, Georges Sand and Marcel Schwob, were used by the discrimination procedure.

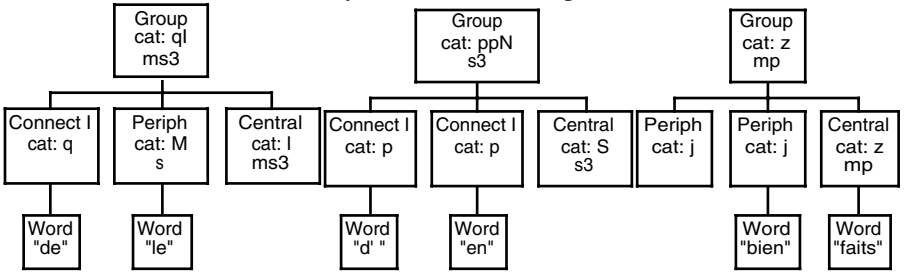

Fig. 5. Three Patterns present in the Lafayette texts without any occurrences in other texts

Among others, the first pattern (cf. fig. 5) covers the following French expression: "*de le supplier*", "*de l'éviter*", "*de l'aimer*", and others like "*de la tromper*", "*à le servir*", "*pour l'obliger*" etc. The second covers "*d'en avoir*", "*d'en attendre*", "*d'en garantir*", "*d'en faire*" etc. but also, "*sans en avoir*" and others which appear to have a very similar structure. While the third covers the following three fragments "*admirablement bien faits*", "*parfaitement bien faits*", "*très bien fait*".
There are many others specific syntactical patterns characteristic of Madame de Lafayette. Among those, here is a syntactical structure which is frequently repeated:

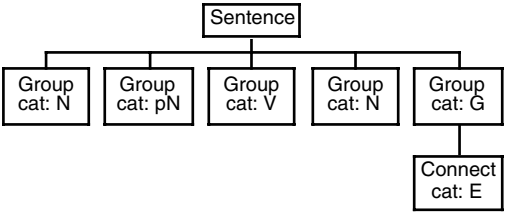

Fig. 6. A syntactical structure characteristic of Madame de Lafayette writings

It closely covers all those fragments (and others in Madame de Lafayette's work) whereas it is virtually absent from the other authors: *"Le prince de Navarre prit la parole :*", "*La reine de Navarre avait ses favorites*", , "*Monsieur de Nemours prit la reine dauphine"*, "*Madame de Clèvcs ne répondit rien*", "*Le comte de Tende aimait déjà le chevalier de Navarre* ;", "*La passion de la reine surmonta enfin toutes ses irrésolutions.*", etc.

There exists also many fragments less closely covered by this pattern. For instance: "*Madame de Chartes avait une opinion opposée*", "*Le comte de Tende sentit son procédé dans toute sa dureté* ;", "*La comtesse reçut ce billet avec joie*", "*L'humeur ambitieuse de la reine lui faisait trouver une grande douceur à régner*", etc.

There are also some very surprising results. For instance, in most of those phrases (more than 80%), the word "*comte*" which means count in French and refers to a member of the aristocracy, is matched against other words like "*prince*", "*madame*" (i.e. madam), "*monsieur*" (i.e. sir), "*reine*" (i.e. queen) and "*comtesse*" (i.e. countess). Since no semantics has been given, this example shows how the syntax may convey semantics. As the reader might well imagine, there are many other hypotheses that may be investigated using this procedure.

All those results were presented to experts of French literature. They recognize some of the pattern as characteristic of the 18th century style of writing, while others seemed to be more specific to Madame de Lafayette. We are now currently integrating our program to a reading environment.

Note that there exists already many Computer-Assisted Research on Literature (CARL) known as stylometric analysis [3], [6]. However, those studies are based on the words, on their repetition, on their size, on their number or on their categories, not on the syntactical structure.

6 Conclusions and Future Research

We have developed a general pattern extraction algorithm working on a SOT. An application to syntactical pattern extraction shows the viability of this algorithm on real life problems. We are now programming a visual interface that displays these results. This program may be applied to many different problems where data are sequential, for instance financial analysis or molecular biology.

Among all the possible applications, one is now under development; it is to analyze traces of computation. Our algorithm is particularly well adapted because traces are easily expressed using a SOT. The two main goals of such an investigation of computing traces are to study the use of new information technology and to detect frequent patterns of commands which are the seeds of new macro operators. In other words, the algorithm can be used to build intelligent learning agents.

References

1. Crochemore M, Rytter W *Text Algorithms*, "Approximate matching", (1994): 237-251.

2. Gusfield D. *Efficient methods for multiple sequence Alignment with Guaranteed Error Bounds*, Bull. Math. Biol., 55 (1993); 141-154.

3. Holmes D Autorship Attribution, *Computers and the Humanities*, 28 (1994): 87-106.

4. Karp R M., Miller R E., Rosenberg A L. *Rapid Identification of Repeated Patterns in Strings, Trees and Arrays*, in Proc. 4th. ACM Symp. Theory of Computing, (1972): 125-136.

5. Landraud A M., Avril J-F, Chrétienne P *An algorithm for Finding a Common Structure Shared by a Family of Strings*, IEEE transactions on Pattern Analysis and Machine Intelligence, 11 (8), (1989): 890-895.

6. Lowe D, Matthews R *Shakespeare Vs. Fletcher: A Stylometric Analysis by Radial Basis Function*, Computer and the Humanities, 29 (1995): 449-461.

7. Rolland, P-Y, Ganascia J-G, *Musical Pattern Extraction and Similarity Assessment.* In Miranda, E. (ed.). Readings in Music and Artificial Intelligence. Contemporary Music Studies - Vol 20. Harwood Academic Publishers. (1999).

8. Sagot, Viari A. A Double Combinatorial Approach to Discovering Patterns in Biological Sequences, *Combinatorial Pattern Matching*, Springer Verlag, LNCS 1075 (1996): 168-208

9. Sankoff D., Kruskal J.B. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, Mass. (1983)

10. Vergne J., *Analyseur linéaire avec dictionnaire partiel, décembre 1999*, convention d'utilisation de l'analyseur de Jacques Vergne. (1999)

11. Zhang K. *Fast algorithms for the constrained editing distance between ordered labeled trees and related problems*, report N°361, Department of computer science, University of Western Ontario, London, Ontario, Canada. (1993)