Neurocomputing 75 (2012) 88–97

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Grammatical inference with bioinformatics criteria

Vivian F. López *, Ramiro Aguilar, Luis Alonso, María N. Moreno, Juan M. Corchado

Departamento Informática y Automática, University of Salamanca, Plaza de la Merced S/N, 37008 Salamanca, Spain

ARTICLE INFO

Available online 4 August 2011

Keywords:
Grammatical inference
Bioinformatics
Context free grammar
DNA
Sequential patterns

ABSTRACT

In this paper we describe both the theoretical and practical results of a novel approach that combines hybrid techniques of association analysis and classical sequentiation algorithms of genomics to generate the grammatical structures of a specific language. We used an application of a compiler generator system that allows a practical application to be developed within the area of *grammarware*, where the concepts of language analysis are applied to other disciplines, such as bioinformatics. The tool allows the complexity of the obtained grammar to be measured automatically from textual data. A technique involving the incremental discovery of sequential patterns is presented to obtain simplified production rules, and compacted with bioinformatics criteria to make up a grammar.

1. Introduction

In recent years many approaches have been introduced as data mining methods for pattern recognition in biological databases. Bioinformatics employs computational and data processing technologies to develop methods, strategies and programs for processing, organizing, and analyzing the immense quantity of biological data.

For this purpose, computational linguistics has received considerable attention in bioinformatics. A study by Searls et al. [32] pointed out that a relation exists between formal language theory and DNA, the linguistic view of DNA sequences being a rich source of ideas for modeling strings with correlated symbols. Most of the work by Jiménez-Montaño et al. [19,20] has involved examinations of grammar complexity and syntactic structure of DNA sequences for the investigation of some current problems in protein structure. Searls et al. [31] found that this kind of linguistic approach proves useful not only in the theoretical characterization of certain structural phenomena in sequences, but also in generalized pattern recognition in this domain, via parsing. The information represented in sequences involves grammatical inference for pattern recognition.

In this work a novel approach is described that combines hybrid techniques of association analysis and classical sequencing algorithms of genomics to generate the grammatical structures of a specific language. Subsequently, these structures are converted to context-free grammars (CFG). Initially the method is applied to context-free languages but it can be applied to other languages: structured programming, the genomic sequences and even the natural languages.

We used the compiler generator tool called GAS 1.0 by López [23], which represents an integrated development environment (IDE) that allows a practical application to be developed within the area for the automatic generation of language-based tools. This practical application is based on traditional solutions and facilitates the use of formal language theory in other disciplines like in grammar-based systems (GBSs) by Mernik [24]. The tool allows the complexity of the obtained grammar to be measured automatically from textual data and thus allows us to verify the effectiveness of the proposed method.

2. Grammars, languages and bioinformatics

2.1. Context-free grammar

A grammar G is defined as $G=(N,T,P,S)$, where N is the set of nonterminal symbols, T is the set of terminal symbols, P is the set of production rules and S is the initial symbol. The language of a grammar $L(G)$ is the set of all terminal strings w that have derivations from the initial symbol. That is:

$$L(G)=\{\text{w is in } T^* | S \Rightarrow^* w\}.$$

A CFG has production rules such as $A\rightarrow\alpha$ where $A\in N$ and $\alpha\in(N\cup T)^*$. The substitution of A by α is carried out independently of the place in which A appears [21]. The majority of the programming languages are generated by grammars of this type (enlarged with some contextual elements necessary for the language semantics).

* Corresponding author.
E-mail address: vivian@usal.es (V.F. López).
URL: http://dptoia.usal.es (V.F. López).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.03.049

2.2. Problem of grammatical inference

Grammatical inference (GI) crosses a number of fields, including machine learning, formal language theory, syntactic and structured pattern recognition, computational biology, speech recognition, and so on, according to Higuera [16].

The problem of GI is the learning of a language description from language data. The problem of context free language inference involves practical and theoretical questions. Practical aspects include pattern recognition; one approach to pattern recognition is the CFG inference that builds a set of patterns, according to Fu [11]; another approach is investigating the ability to infer CFGs from nature, which would enable a speech recognizer to modify its internal grammar on the fly, thus allowing it to adjust to individual speakers. Theoretical aspects are important in regard to the serious limitations of context-free languages, following Lee [22] and actually in constructing feasible algorithms of learning that imitate the human language model (this last point can be problematic, but was one of the principal motivations for the early work in grammar inference) [17].

2.3. Language learning

The learning of a language also has to do with its identification. In the GI literature, attention is focused on the *identification in the limit*, where, in each time t the machine that is learning receives an information unit on a language and puts out a hypothesis $H(i_1,\ldots,i_t)$; the learning algorithm is successful if after a finite amount of time, all its guesses are the same and are all a correct description in the language in question. Another learning criterion is the *exact identification using queries in polynomial time*; in this framework, the learning machine has access to the oracles that can answer questions, and must halt in polynomial time with a correct description of the language [22].

In the last 20 years, the inherent complexity present in the problem of GI has rendered all the approaches unsuccessful, according to Miclet [25]. The paper detailed by Higuera [15] states that actually the algorithms with mathematical properties obtain better results than the algorithms with heuristic properties, but when finite automata are used or, alternatively, when algorithms of GI learning are constructed. It also emphasizes that a common *benchmark* does not exist for the heuristic approach and it is thus more difficult to compare and to evaluate the effectiveness of these methods. With these points in mind, our approach is tested on an actual data set, which we can share with other researchers upon request to improve or ratify our work.

2.4. Grammars and bioinformatics

Bioinformatics employs hybrid techniques [1,9] to handle, order and study the immense quantity of biological data that have been generated and are currently being generated. For example, for the human genome (HG), bioinformatics seeks to find meaning in the language of the more than 37000 million pairs of A, C, T and G that have been compiled and stored in the *book of life*.

It offers us the opportunity to understand the gigantic database (DB) that contains the details of the circumstances of time and place in which the genes are activated, the configuration of the proteins that they specify, the way in which they influence certain proteins or others and the role that such influences can play in diseases. It can also help us to tackle questions such as how the HG is related to the genomes of the model organisms, like fruitflies, mice and bacteria; the possibility that it will be able to discover sequential patterns that show how information fragments are related to each other; or if it will it be able to make up a grammatical structure that shows the interpretation of the resultant set. If we are able to infer that structure for this type of language we will be contributing to an understanding of the real function of the structure of DNA.

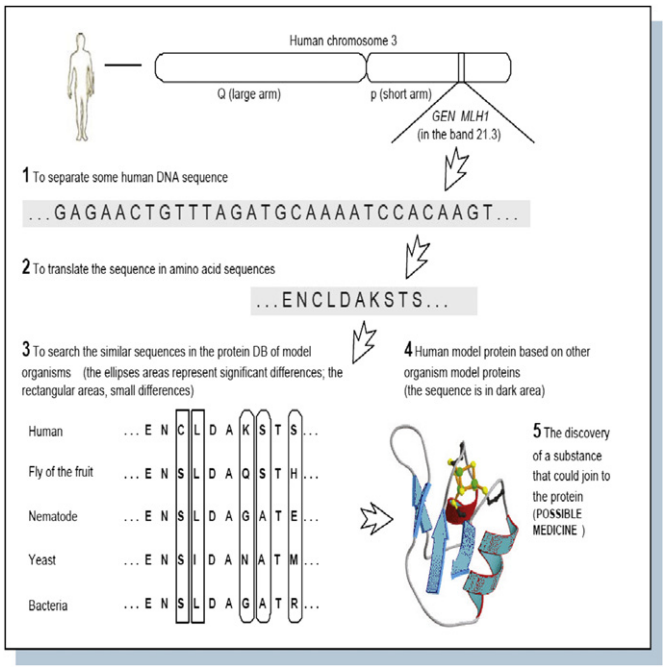


Fig. 1. Use of bioinformatics in pharmacology.

One of the applications of bioinformatics is in pharmacology, where it offers new solutions to the old model for the creation of new medicines. It is worth noting that one of the most elementary bioinformatics operations is searching for resemblances between a recently arranged fragment of DNA and the already available segments of diverse organisms (it remembers and associates this with the discovery of sequential patterns). The finding of approximate alignments allows it to predict the type of protein that will specify such a sequence. This not only provides trails in pharmacological designs in the initial phases of drug development, but also suppresses some which will end up being a puzzle to be solved. A popular series of programs to compare sequences of DNA is the basic local alignment search tool (BLAST) [4,5], whose mechanism of comparison applied in the development of new medicine is shown in the flow chart in Fig. 1.

3. Techniques for the association analysis

Association analysis involves techniques that differ in their operations, but all of them search for relations among the attributes of a data set. Some techniques are:

1. Association rules.
2. Discovery of associations.
3. Discovery of sequential patterns.

3.1. Association rules

Association rules (AR) describe the relations of certain attributes with regard to other attributes in a DB. These rules identify cause-effect implications between the different attributes of the DB. For example, in the registers of product purchases, it can tell us which article of purchase is identified as related to another; for

example: "Eighty percent of the people who buy diapers for a baby also buy talcum powder".

A rule takes the form

if X then Y or $X \Rightarrow Y$

X is called the *antecedent of the rule* (in the example, "buy diapers"); Y is called the *consequent* of the rule (in the example, "buy talcum powder").

The generation of the rule is supported by statistical and probabilistic aspects such as the support factor (f_s), confidence factor (f_c) and the expected confidence factor (f_e) defined as

$$f_s = \frac{nr\_times\_rule}{nr\_total\_registers}$$

$$f_c = \frac{nr\_times\_rule}{nr\_times\_X}$$

and

$$f_e = \frac{nr\_times\_Y}{nro\_total\_register}$$

The minimum value of the support factor for the rules should be greater than a given threshold. If the confidence factor is greater than 0.5, then the rule appears in at least half the number of instances, meaning that the rule makes certain sense. The difference between the support factor and the expected confidence factor should be minimal to ensure the effectiveness of the rule.

For example, if we consider the data in Table 1, a rule obtained is: $A=2 \Rightarrow B=2$ with $f_s=0.43$, $f_c=0.75$ and $f_e=0.57$, which means that 75% of the items with $A=2$ imply $B=2$, and moreover, 43% of all the items comply with that rule and $B=2$ complies with 57% of all the items.

Table 1
Data set for association rules.

A	B	C	D	E	F
2	2	6	0	1	0.2
2	2	5	0	1	0.2
2	2	6	1	1	0.2
3	2	7	1	0	0.8
2	3	8	1	0	0.8
3	3	8	1	0	0.8
3	3	7	1	0	0.8

Association rules have been widely used in bioinformatics in gene expression data analysis, Cai et al. [7]. Although association rules have been widely used, there is no systematic way to select interesting rules from the millions generated from high dimensional gene expression data. In this study, a kernel density estimation-based measurement is proposed to evaluate the interest of the association rules. Several pruning strategies are also devised to efficiently discover the approximate top-k interesting patterns. Finally, the overfitting problem of the classification model is addressed by using the conditional independence test to eliminate redundant rules. Experimental results show the effectiveness of the proposed interest measure and classification model.

3.2. Discovery of associations

In a way similar to the AR, the discovery of associations (DA) tries to find implications between the attribute values of different couples such that their appearance determines a present association in a good amount of the DB entries. To discover associations the following steps are carried out:

1. Associate an identifier with each transaction.
2. Order the transactions sequentially according to their identifier.
3. Count the occurrences of the articles creating a vector where each article is counted. The elements for which the count is below a *threshold* are eliminated.
4. Combine in a matrix the transactions' attribute values and count the number of occurrences, eliminating those elements that do not surpass the threshold.
5. Repeat steps 3 and 4 successively until no more transaction combinations are possible. With the data from Table 1, with threshold=2, the technique is applied as shown in Fig. 2 and the following associations are generated:
   (a) $A2,B2,E1,F0.2 \rightarrow class\ 1$
   (b) $A3,D1,E0,F0.8 \rightarrow class\ 2$
   (c) $B3,D1,E0,F0.8 \rightarrow class\ 3$

Association (a) means: if the value of A and B is 2 and the value of E is 1 and the value of F is 0.2, then the entries with those characteristics can belong to a class. The other associations show the possible characteristics of the entries for belonging to another class or behavior.

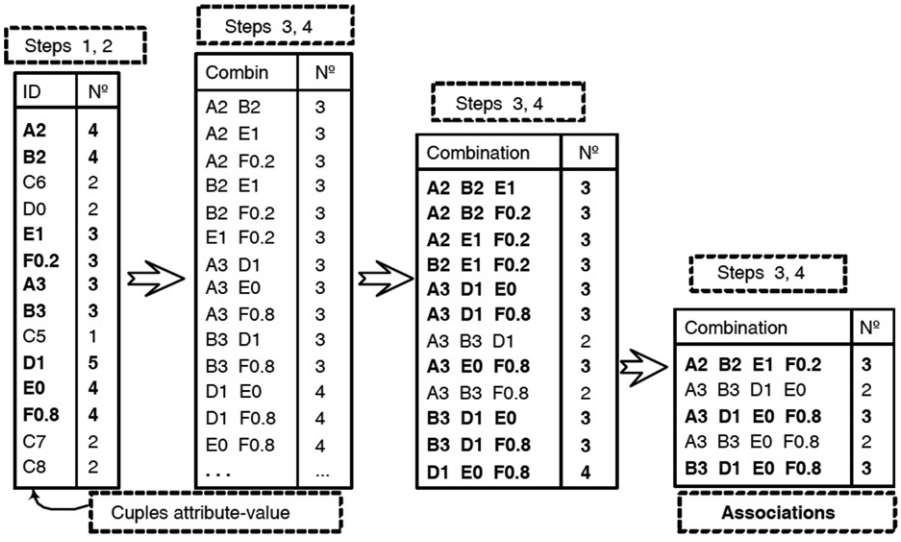


Fig. 2. Discovery of associations in data from Table 1.

A novel invariant pattern recognition approach is proposed by Ciocoiu [8] based on a special gradient-type recurrent analog associative memory. The system exhibits stable equilibrium points in predefined positions specified by feature vectors extracted from the training set, while invariance to geometrical transformations is inferred by using the tangent distance. Experimental results for handwritten character recognition and face recognition tasks indicate that the proposed approach may yield superior performances over classical solutions based on the Euclidean distance metric. Possible extensions towards modular and sequential pattern recognition are finally outlined.

3.3. Discovery of sequential patterns

Discovery of sequential patterns (DSP) is very similar to AR but searches for patterns between transactions so that the presence of a set of items precedes another set of items in a DB during a period of time. For example, if the data correspond to entries of articles purchased by clients, a description of what articles a client buys can frequently be obtained, and above all, what the sequence of his or her purchase is. Thus, the next time, the profile of the client would be known, and it will be able to predict the sequence of his or her purchase. This criteria can be applied to another data control, for example, in the bioinformatics context, when the data to be processed correspond to the chain of nucleotides of the genome and sequences are discovered as the patterns that code genes make up some protein [2,10]. DSP has the following operation:

1. Identify the time-related attribute.
2. Considering the period of time when the sequential patterns are to be discovered, create an array ordered by the identifier of the transaction.
3. Create another array linking the articles of purchase of each client.
4. Infer the sequential patterns according to the *support percent*.

The patterns discovered show instances of articles that appear in consecutive form in the data as can be seen in the example in Fig. 3 (drawn up according to [6]). Pei et al. [27] and Zaki [36] proposed methods that efficiently discover frequent sequential patterns based on their support from discrete sequential data. They used a well-known criterion that evaluates their frequency. Garofalakis et al. [12] have presented a method that discovers sequential patterns satisfying regular expressions specified by the analysts. Sakurai et al. [30] proposed a new sequential pattern mining method. It introduces a new evaluation criterion to replace the previous evaluation criteria satisfying the *A priori property*. The criterion is calculated by the frequency of the sequential pattern and the minimum frequency of items included in the items. It extracts sequential patterns that can be rules for predicting future items with high probability. Also, the method introduces new constraints. The constraints extract item sets composed of items whose attributes are different and extracts sequential patterns composed of item sets whose attribute sets are equal to one another. The mining method based on the criterion and the constraints can efficiently discover sequential patterns coinciding with analysts' interests.

4. Association analysis-based procedure for grammatical inference

We consider the experiences acquired by Aguilar [2], the literature and the existing theories [21,13,26], in order to process data that are not structured in relations or tables with differentiated attributes but are coded as a finite succession of sentences. The procedure includes the following phases:

1. Language generation by means of a CFG. This language will be the data source.
2. Coding of the strings of the language regarding its syntactic categories.
3. Disregarding the initial grammar, discover the sequential patterns in the coded language. This discovery, called incremental, is a combination of the operation of the DSP and of the operation of the search for identical sequences in bioinformatics [10]. With this, sequential patterns will be found and replaced by an identifier.
4. Replace the discovered sequences with their identifiers. This way the identifier and the sequence are stored as a production rule.
5. Repeat the two previous steps until all the sentences of the language are replaced by identifiers.

4.1. Language generation

We consider the *CFG*, $\mathcal{G}_{æ}$ proposed by Louden [21] in the generation of arithmetic expressions; the majority of the

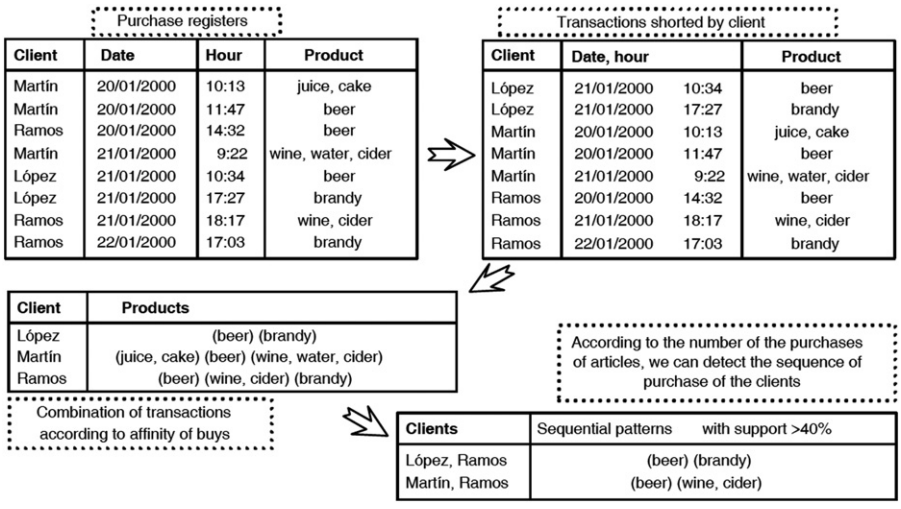


Fig. 3. Discovery of sequential patterns in registers of product purchases.

Table 2
Modification of the $\mathcal{G}_{æ}$*Grammar*.

$\mathcal{G}_{æ}$*Grammar*	$\mathcal{G}_{æ}$*Grammar*
$G_{æ}=(N,T,P,S)$	$\mathcal{G}_{æ}=(\mathcal{N},\mathcal{T},\mathcal{P},S)$
$N=\{Exp,Num,Dig,Op\}$	$\mathcal{N}=\{E,d,b,o,a,c\}$
$T=\{0,1,+,*\}$	$\mathcal{T}=\{0,1,+,*,(,)\}$
$P: Exp\rightarrow Exp\ Op\ Exp \mid (Exp) \mid Num$	$P: E\rightarrow E\ o\ E \mid aEc \mid n$
$Num\rightarrow Dig^{+}$	$d\rightarrow b^{+}$
$Dig\rightarrow 0 \mid 1$	$b\rightarrow 0 \mid 1$
$Op\rightarrow + \mid *$	$o\rightarrow + \mid *$
	$a\rightarrow ($
	$c\rightarrow)$
$S=Exp$	$S=E$

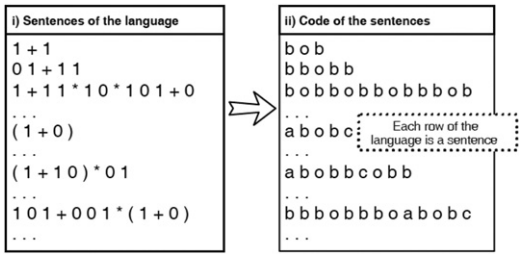


Fig. 4. Language of arithmetic expressions on which its grammar inferred.

programming languages are generated by grammars of this type. We can modify the formalism of this *CFG* in the following way (Table 2) to add a new syntactic construct to specify and search DNA patterns in data. A DNA molecule can then be represented as a finite string of symbols from this alphabet; a language, formally, is any set of such strings [31]. The new grammar created, $\mathcal{G}_{æ}$, does not change in essence the character of the original grammar.

With the previous criteria, a sample of the language generated by $\mathcal{G}_{æ}$ can be seen in Fig. 4, point (i). Note that each line corresponds to a sentence accepted by the grammar.

4.2. Language coding

Considering the language that is generated with $\mathcal{G}_{æ}$, all the symbols of $\mathcal{T}$ can be coded with the symbols of $\mathcal{N}$. For this particular case the symbols to be used are $\{b,o,a,c\}$ as syntactic categories. See Fig. 4, point (ii).

4.3. Incremental discovery of sequential patterns and associations

We have developed a new DSP algorithm which calculates the global scoring (g_s) of candidate sequential patterns and selects the largest overall scoring pattern. The algorithm builds a weighting scheme to decide the next production rules to be constructed.

The key element in the algorithm is to optimize the total size of the grammar and generate production rules for GI. The algorithm accepts as input a set of sentences of language *Q*, defined as a string of length W_Q, and the outputs are production rules.

The DSP algorithm is applied to coded languages with the aim of finding key subsequences in the sentences of the language. Each subsequence s has a length L_s that indicates the number of symbols included in Q. Where n_s represents the appearances of the number of s in *Q*, we used the Boyer–Moore–Horspool algorithm [18] to calculate n_s. In this particular case $1\leq L_s\leq 5$.

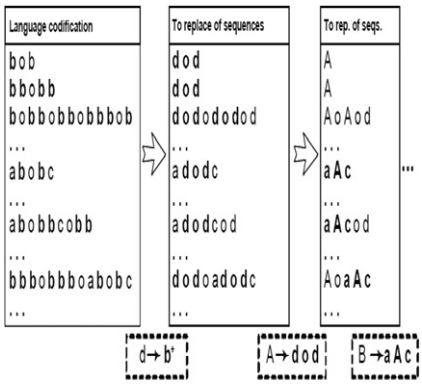


Fig. 5. Hybrid discovery of sequential patterns for the context-free languages.

To reduce the computational cost of the experiments we also used the Boyer–Moore–Horspool algorithm to empirically obtain the pattern length, noting that for a *pattern length* ≤ 5 we will guarantee a suitable internal representation of the data. By convention, in the coded language many sentences exist that make up the population of the language. The idea consists in finding subsequences, identifying them with a symbol and replacing the appearances of the subsequences in the sentences of the population with that symbol, repeating the procedure until each sentence is identified by a single symbol (see Fig. 5).

The detailed steps of the general algorithm can be found in [3]. Below is the pseudo-code of the proposed algorithm.

Algorithm 1. Incremental discovery of sequential patterns.

1: **Input**: *Q* a language sentence test set of length W_Q
2: **Output**: production rules
3: initialize $g_s=0$
4: **for** all the sentences *Q* **do**
5: **while** $W_Q\geq 1$ **do**
6: **for** $L_s=1$ to 5 **do**
7: make each subsequence **s** from $\mathbf{L_s}$ first symbols of **Q**.
8: compute the global scoring $\mathbf{g_s}$ of **s** defined as $\mathbf{g_s}=\sum_{i=1}^{quant}\mathbf{p_s^i}$, where $\mathbf{P_s}=\frac{\mathbf{L_s}*\mathbf{n_s}}{\mathbf{W_Q}}$, is the scoring of s in Q and *quant* is the quantity of candidate sequential patterns.
9: **end for**
10: select the subsequence **s*** of greatest overall scoring
11: **if s*** has one symbol **then**
12: replace all the consecutive appearances of that symbol by itself. Thus the production rule is created $\alpha\rightarrow\beta^{+}$ (in this particular case, all the *bb* are replaced by *d*, see Fig. 5).
13: **else**
14: replace all the appearances of **s*** in the sentences **Q** creating the production rule $A\rightarrow$*contained_in_***s***. The symbol *A* is generated consecutively so that the next time that another rule production is created, *B*,*C*, …is utilized and so on (see Fig. 5).
15: **end if**
16: **end while**
17: **end for**
18: return to step 1 noting that with 11 and 14 the size of the sentences of the language population changes.

With the above procedure, production rules are generated that recognize the sentences of the language. The number of production rules can be considerable, so we apply a particular method of simplification.

5. Experiments

5.1. Rule similarity

Considering the language $\mathcal{L}_{æ}$ of arithmetic expressions, the production rules of Fig. 6 were obtained by applying the DSP algorithm.

With the right hand side of the production rules, which form the sequential patterns of the language, a *substitution matrix* is computed, which shows the similarity values between terminal symbols, as shown in Fig. 7.

Similarity between a pair of consecutive symbols is related to the frequency of the symbols in the language (like the BLOSUM matrix) [14]. Subsequently, it is possible to make alignments among those sequences by compacting them. We have used the *estimator Scoredist* proposed by [33] to determine the distance between a terminal and a nonterminal which uses a logarithmic correction of observed divergence based on the alignment score according to the BLOSUM score matrix and its main advantages are computational simplicity and high robustness.

In the substitution matrix $m(i, j)$, each row i and each column j corresponds to a nonterminal symbol of the production rules generated. The values of the matrix denote the importance of the alignment among the nonterminal symbols; for example, $m(d, d)=23$ denotes a high degree of similarity between both symbols; $m(d, A)=-1$ denotes a degree of similarity of -1.

The symbols are ordered by frequency, as shown in Table 3, that is first *d*, *A*, *C*, *o* and so on. For $\mathcal{L}_{æ}$ it generated 19 symbols *A*,*B*,...,*S* that join with the symbols of the coding *d*, *o*, *c* and *a* and make up 23 nonterminal symbols (in the bioinformatics context, the symbols would correspond to the amino acids).

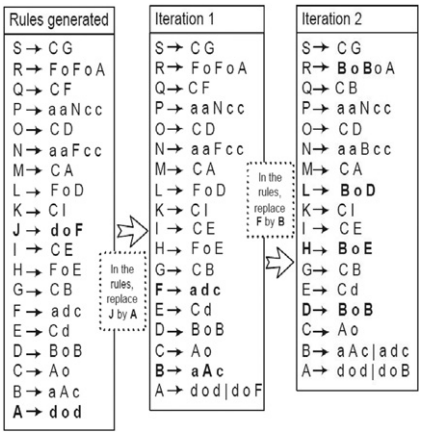


Fig. 6. Production rules generated and some iterations in their simplification.

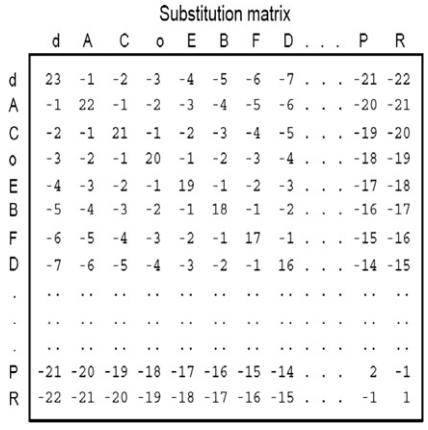


Fig. 7. Substitution matrix for the rules generated.

Table 3
Frequency of subsequences for the language $\mathcal{L}_{æ}$.

Symbol	Value	Freq.
d	b^+	0.7500
o	$+\|*$	0.1704
a	(	0.0398
c	)	0.0398
A	*dod*	0.4321
B	*aAc*	0.1115
C	*Ao*	0.1725
D	*BoB*	0.0696
E	*Cd*	0.1128
F	*adc*	0.0746
G	*CB*	0.0532
H	*FoE*	0.0311
I	*CE*	0.0519
J	*doF*	0.0195
K	*CI*	0.0311
L	*FoD*	0.0195
M	*CA*	0.0195
N	*aaFcc*	0.0061
O	*CD*	0.0195
P	*aaNcc*	0.0039
Q	*CF*	0.0156
R	*FoFoA*	0.0039
S	*CG*	0.0156
T	*CB*	0.0195

5.2. Rules simplification and compaction

With the right hand parts of the production rules (where the first rules generated are more important) we search for *similar sequences* to compact them. The steps are:

1. The sequence β that can be compacted with the sequence α is activated with the similarity function f;

$$f(\alpha,\beta)=\begin{cases}1 & \text{if } \dfrac{\sum_{i=1}^{n} m(\alpha_i,\beta_i)}{\sum_{i=1}^{n} m(\alpha_i,\alpha_i)}\geq \Theta,\\ 0 & \text{otherwise,}\end{cases}$$

   where n is the minimal length between the sequences α and β, and Θ is a threshold or *similarity factor*, with value 0.4 in this particular case.
2. The similar sequences are compacted and will be derived by a single nonterminal symbol. The remaining nonterminal symbol should be replaced by the previous one in all the right parts of the rules.
3. Repeat the previous steps until there are no similar sequences.

For example, for the language $\mathcal{L}_{æ}$:

The rules **dod** and **aAc** are not similar since $f(\mathbf{dod},\mathbf{aAc})=0$ since

$$\frac{\sum m(\mathbf{dod},\mathbf{aAc})}{\sum m(\mathbf{dod},\mathbf{dod})}=\frac{-10-2-11}{23+20+23}=-0.35$$

is not greater than 0.40.

Nevertheless, the rules **dod** and **doF** are similar since,

$$\frac{\sum m(\mathbf{dod},\mathbf{doF})}{\sum m(\mathbf{dod},\mathbf{dod})}=\frac{23+20-6}{23+20+23}=\frac{37}{66}=0.56.$$

This way, the generated rules are simplified and compacted iteratively (Figs. 6 and 7) until a grammar $\mathcal{G}'_{æ}$ is built. The grammar $\mathcal{G}'_{æ}$ is described in Table 4. The system forms new grammar rules from repeated sequences, and also merges rules

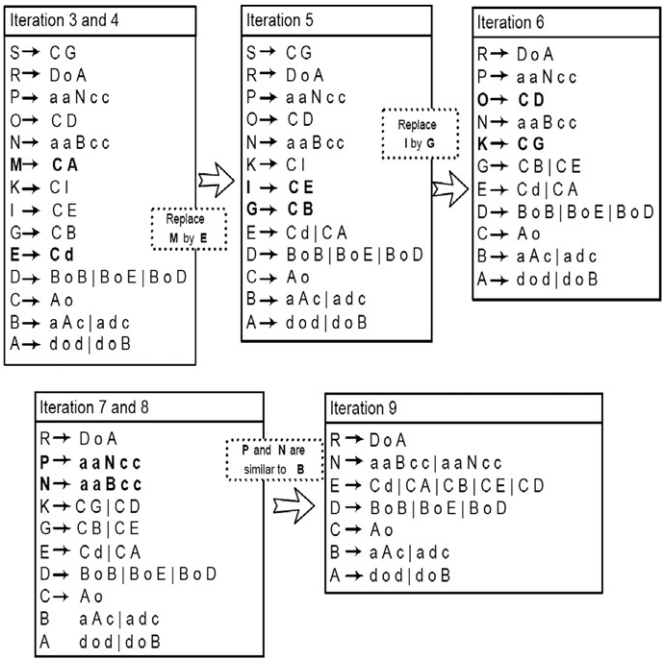


Fig. 8. Simplification and compaction of the production rules generated.

Table 4
The generated grammar $\mathcal{G}'_{æ}$.

$\mathcal{G}'_{æ} = (\mathcal{N}', \mathcal{T}', \mathcal{P}', \mathcal{S}')$
$\mathcal{N}' = \{S,R,E,D,B,A,d,b,o,a,c\}$
$\mathcal{T}' = \{0,1,+,*,(,)\}$
$\mathcal{P}' : S \to R|E|D|B|A|d$
$R \to DoA$
$E \to Cd|CB|CE|CA|CD$
$D \to BoB|BoE|BoD$
$C \to Ao$
$B \to aAC|adc$
$A \to dod|doB$
$d \to b^+$
$b \to 0|1$
$o \to +|*$
$a \to ($
$c \to)$
$\mathcal{S}' = S$

Table 5
similarity matrix *M*.

	a	*b*	*c*	*d*	*e*	*f*	...
a	0	1	2	3	4	5	...
b		0	1	2	3	4	...
c			0	1	2	3	...
d				0	2	3	...
e					0	1	...
f						0	...
..							...

to generalize grammars, because they expect a set of sentences drawn from a language, rather than a single long sequence, so that the existence of sequences produced by the grammar that are not part of the input sequences does not affect the grammar generation. The study by Stolcke and Omohundro [34] indicates that the performance of GI algorithms is measured by their ability to accept test sentences from the language, and to reject new sentences that are not in the target languages.

5.3. Automatic generation of syntactic categories

Considering language $\mathcal{L}_{æ}$ as a data source (corpus) and also supposing that we do not know the syntactic categories, the algorithm must assume that language $\mathcal{L}_{æ}$ is coded with as many symbols as words are in the language.

The proposed algorithm can be summarized as follows:

Algorithm 2. Automatic discovery of syntactic category.

1: **Input**: a corpus
2: **Output**: syntactic category
3: **for** all corpus **do**
4: initialize a similarity **matrixM**.
The *matrixM* shows the similarity values between initial codes, depending on the generation order (see Table 5).
5: replace all the consecutive symbols by only one.
6: let **A** be the subsequence with the highest frequency and the highest size possible.
7: find each subsequence *A*.
8: **where**
9: the **size** is calculated for each symbol between 1 and $Min(|A|,3)$,
10: $|\mathbf{A}|$ is the number of symbols s_i of *A*,
11: $\mathbf{f_A} = \frac{1}{\mathbf{quant}} \sum_{i=1}^{quant} \frac{\mathbf{A_{s_i}} * |\mathbf{A}|}{\mathbf{s_i}}$ is the frequency of *A*,
12: **quant** is the number of sentences of the language,
13: $|\mathbf{s}|$ is the symbol number of each sentence.
14: **if** the frequency of $A \ll \Theta$ **then**
15: **End**
16: **else**
17: join the symbols generated consecutively, the highest frequency subsequences with the lowest frequency subsequences.
18: for all *A* sequences find all similar subsequences, (*B*,*C*, ..., and so on).
19: With the subsequences *A*,*B*,*C*, ..., make multiplex alignments among them to find a consensus sequence **X**.
20: **if** $A=ab$ $B=ac$ $X=ab$ **then**
21: join the symbols of *c* with the symbols of *b* and replace all the appearances of *c* by *b* in the coded language.
22: **end if**
23: **if** $A=abc$ $B=acd$ $C=abe$ $X=abd$ **then**
24: join the symbols of *c* and *e* with the symbols of *d* and replace all the appearances of *c* and *e* by *d* in the coded language, as in [29].
25: **end if**
26: **end if**
27: **end for**

The automatic generation of syntactic categories method was applied to language $\mathcal{L}_{æ}$ of arithmetic expressions, and the steps carried out can be seen in Fig. 9.

At the beginning the language was coded with six symbols *a*, *b*, *c*, *d*, *e* and *f*; there are as many symbols coded as different symbols in the language. Each symbol is the tag of the syntactic category that has a single symbol. Later, $\mathbf{a}^+$ consecutive patterns be replaced by **a**, and all the *b* by *a* (because the most frequent element is *a*). We find **aca** sequences and **ada** sequences to align them, then we infer the union of the elements of the syntactic category **d** and the syntactic category **c** and replace the symbol **d**

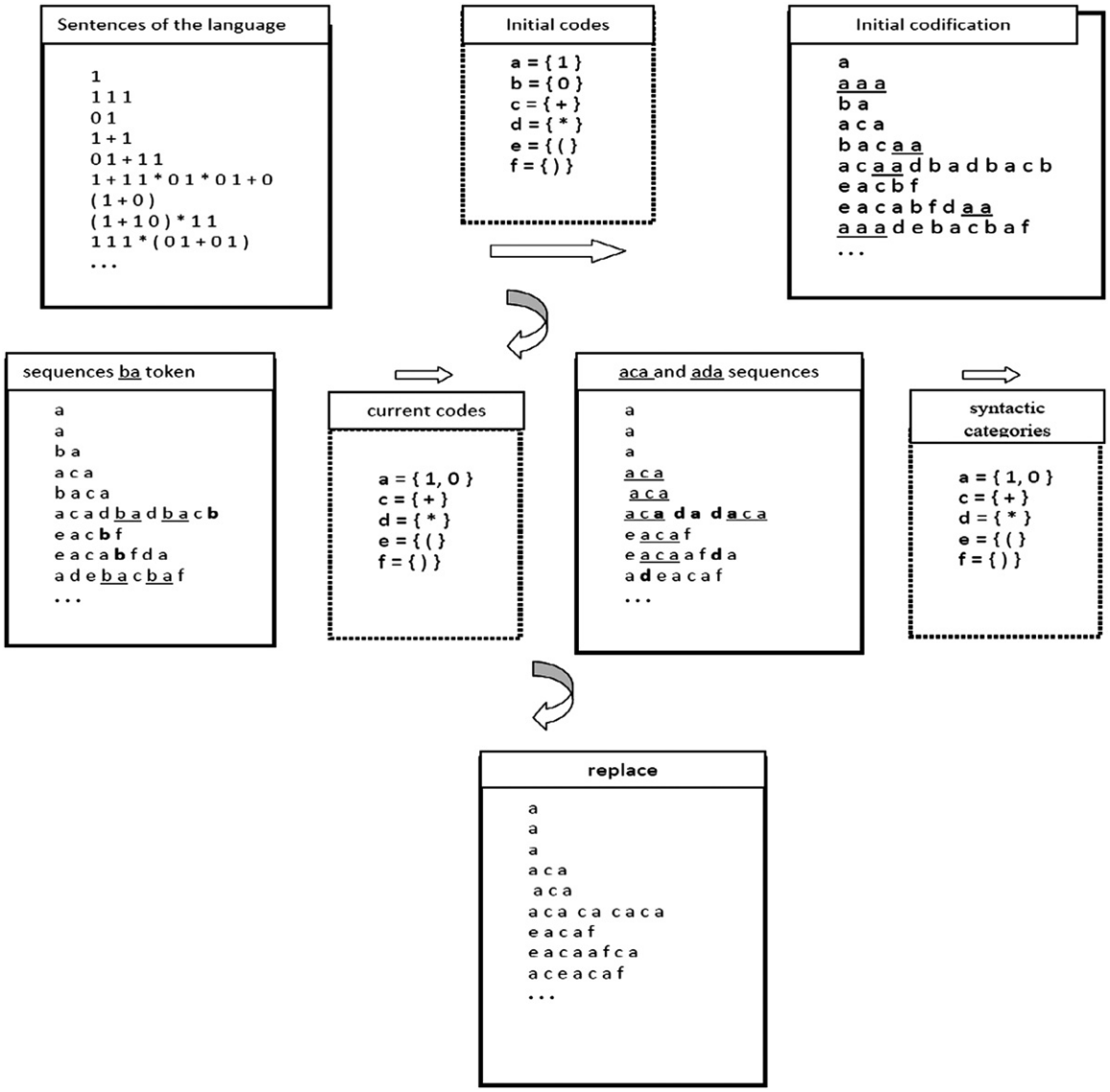


Fig. 9. Automatic generation of syntactic categories method.

in the language by **c**. Finally four syntactic categories are created which are labeled with the symbols *a*, *c*, *e* and *f*.

6. Practical results using GAS 1.0

GAS 1.0 by López [23] provides a basis to create new components in application fields possibly different from the traditional ones, more precisely in data mining for the discovery of biological data. With this goal in mind, languages like $\mathcal{L}_{æ}$ created with grammar $\mathcal{G}'_{æ}$ were considered for automatic design methods to generate analyzers and/or language translators that can facilitate this task of parsing a string. In this respect, we used the compiler generator GAS 1.0 to automatically generate a scanner and a parser for the language specification. Taking as input the grammar specification $\mathcal{G}'_{æ}$, the syntactic analysis tables are created, giving as a result a Decorated Abstract Syntax Tree (DAST), or the syntactic error. The DAST reflects the grammar rules applied and gives a kind of structural description of grammatical features in the input string, exactly the kind of output that is desired in describing certain biological sequence data [31]. We used a measurement of complexity of the structure of the grammars that will provide a method to evaluate the syntactic framework of the grammars $\mathcal{G}'_{æ}$. The definition of the measurement comes from the concepts described by López [23], which will be used in the objective evaluation of the quality of the grammars:

- Number of nonterminals: This allows the size of a CFG to be measured, by applying a *fine degree metric* whose use in the evaluation of the complexity of programs is focused on the number of procedures.
- Cyclomatic complexity: Defined by Pressman [28] and Wallace [35], McCabe's complexity, or the cyclomatic complexity V of a flow graph G, is defined. This complexity is defined as follows:

  $$V(G) = A - N + 2$$

  where A is the number of edges of the flow graph and N the number of nodes.

These metrics have been implemented in the tool. The measurement of the number of nonterminal elements is trivial. With regard to the cyclomatic complexity, in order to facilitate its understanding the associated graph is constructed. Our approach confirms the idea that

the grammar complexity has been applied successfully. It also confirms the effectiveness of the proposed method. For example, for the grammars $\mathcal{G}'_{æ}$, the complexity is on the order of 9, which represents the minimum of all complexity of the sequences computed. Its high values confirm that the obtained grammars are good, and they can offer the best results for the analysis of biosequences, providing sufficient discrimination.

We propose a different practical approach in this paper to the problem of GI, which, to the best of our knowledge, has not been discussed in the literature before. As a first step, we apply the incremental discovery of sequential patterns to obtain simplified production rules, and these are compacted with bioinformatics criteria to make up a grammar. Then, we apply a measurement of the complexity of the structure of the grammar which will be used in the objective evaluation of the quality of the grammar. We argue that this approach could be more beneficial than the others, as it considers the complexity of the obtained grammar. In order to evaluate our hypothesis and verify the effectiveness of the proposed method we compared the complexity. We also created a new benchmark for comparing and evaluating GI in heuristic approaches. Therefore, the proposed method can be used to compare and to evaluate the effectiveness of GI and empirically performs better than the existing algorithms.

7. Conclusions

In the experiments, a language $\mathcal{L}_{æ}$ generated by predetermined CFG $\mathcal{G}'_{æ}$ is considered. But later none of the properties of that grammar were utilized to generate the set of production rules that then made up the grammar $\mathcal{G}'_{æ}$. We have proposed a new method of automatic generation of syntactic categories in a coded language. The approach can be extended to the processing of data that are believed to have a grammatical structure that could be automatically generated.

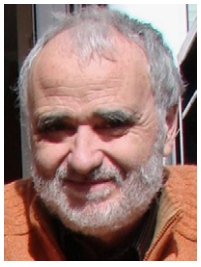
The algorithm can be applied in different fields, and we can imagine finding something similar for the analysis of biosequences or for the natural languages. The IDE attenuates the complexity of the design of the grammar specification, improves the quality of the obtained product and sensibly diminishes the development time and cost. We have tried to reduce the learning time for nonexpert users in the area of compiler generation. The tool allows the complexity of the obtained grammar to be measured automatically from textual data and thus permits verification of the effectiveness of the proposed method.

References

[1] A. Abraham, E. Corchado, J.M. Corchado, Hybrid learning machines, Neurocomputing 72 (13–15) (2009) 2729–2730.
[2] R. Aguilar, Data Mining, Foundations, technologies and applications, Salamanca University, Salamanca, Spain, 2003.
[3] R. Aguilar, Incremental discovery of sequential patters for grammatical inference, Ph.D. Thesis, Salamanca University, Spain, 2007.
[4] S.F. Altschul, W. Gish, W. Miller, E.W. Meyers, D.J. Lipman, Basic local alignment search tool, Molecular Biology 215 (1990) 403–410.
[5] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman, Gapped blast and psi-blast: a new generation of protein database search programs, Nucleic Acids Research 25 (1997) 3389–3402.
[6] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, A. Zanasi, Discovering Data Mining. From Concept to Implementation, Prentice-Hall, 1998.
[7] R. Cai, Z. Hao, W. Wen, H. Huang, Kernel based gene expression pattern discovery and its application on cancer classification, Neurocomputing (2010) 2562–2570.
[8] I.B. Ciocoiu, Invariant pattern recognition using analog recurrent associative memories, Neurocomputing 73 (1–3) (2009) 119–126.
[9] E. Corchado, A. Abraham, A.C. Ponce Leon Ferreira de Carvalho, Hybrid intelligent algorithms and applications, Information Science 180 (14) (2010) 2633–2634.
[10] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, MIT Press, 1996.
[11] K.S. Fu, Syntactic Methods in Pattern Recognition, Academic Press, 1974.
[12] M.N. Garofalakis, R. Rastogi, K. Shim, Spirit: Sequential pattern mining with regular expression constraints, in: The 25th International Conference on Very Large Data Bases, September 1999, pp. 223–234.
[13] R. Hausser, Foundations of Computational Linguistics: Human-Computer Communication in Natural Language, Springer, Berlin, 2001.
[14] S. Henikoff, J.G. Henikoff, Amino acid substitution matrices from protein blocks, Proceedings of the National Academic Science 89 (1992) 10915–10919.
[15] C. Higuera, Current trends in grammatical inference, in: Proceedings of Joint Iapr International Workshops Spr 2000 and Spr 2000. Lecture Notes in Artificial Intelligence, Springer-Verlag, 2002, pp. 130–135.
[16] C. Higuera, A bibliographical study of grammatical inference, Pattern Recognition (2004).
[17] J.J. Horning, A study of grammatical inference, Technical Report 139, Computer Science Department, Stanford University, 1969.
[18] R.N. Horspool, Practical fast searching in strings, Software-Practice & Experience 10 (6) (1980) 501–506. doi:10.1002/spe.4380100608.
[19] M.A. Jiménez-Montaño, On the syntactic structure of protein sequences and the concept of grammar complexity, Bulletin of Mathematical Biology 46 (4) (1984) 641–659.
[20] M.A. Jiménez-Montaño, H.R. Lucio-García, A. Ramos-Fernnádez, Computer simulation to generate simplified proteins with stochastic grammars, Periodicum Biologorum 107 (4) (2005) 397–402.
[21] K.C. Louden, Compiler Construction, Principles and Practice, International Thomsom Publishing Inc., 1997.
[22] L. Lee, Learning of context-free languages: a survey of the literature, Technical Report, Computer Science Department, Stanford University, 1996.
[23] V. López, A. Sánchez, L. Alonso, M.N. Moreno, A tool to create grammar based systems, in: Advances in Soft Computing, volume 50/2009, Springer, Berlin, Heidelberg, 2009, pp 338–346. ISBN: 978-3-540-85862-1.
[24] M. Mernik, M. Crepinsek, T. Kosar, D. Rebernak, V. Zumer, Grammar-based systems: definition and examples, University of Maribor, 2004.
[25] L. Miclet, Structural Methods in Pattern Recognition, Chapman and Hall, 1986.
[26] S. Mitra, T. Acharya, Data Mining, Multimedia, Soft Computing and Bioinformatics, John Wiley and Sons, 2003.
[27] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M.C. Hsu, Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth, in: The 17th International Conference on Data Engineering, April 2001, pp. 215–224.
[28] R.S. Pressman, Software Engineering: A Practitioner's Approach, McGraw Hill, New York, 2002.
[29] Y. Sakakibara, Efficient learning of context-free grammars from positive structural examples, Information and Computation 97 (1992) 23–60.
[30] S. Sakurai, Y. Kitahara, R. Orihara, K. Iwata, N. Honda, T. Hayashi, Discovery of sequential patterns coinciding with analysts interests, Journal of Computers 3 (7) (2008). doi:10.4304/jcp.3.7.1-8.
[31] D.B. Searls, S. Dong, A syntactic pattern recognition system for DNA sequences, in: International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis, 1993, pp. 89–101.
[32] D.B. Searls, et al., Formal language theory and biological macromolecules, 1999, available at: ⟨http://citeseer.ist.psu.edu/searls99formal.html⟩.
[33] E.L.L. Sonnhammer, V. Hollich, Scoredist, a simple and robust protein sequence distance estimator, BMC Bioinformatics 6 (108) (2005). doi:10.1186/1471-2105-6-108.
[34] A. Stolcke, S. Omohundro, Inducing probabilistic grammars by Bayesian model merging, in: Proceedings of the Second International Conference on Grammatical Inference and Applications, Springer-Verlag, Alicante, Spain, 1994, pp. 106–118.
[35] D.R. Wallace (Ed.), Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, NIST SP 500-235, August 1996.
[36] M.J. Zaki, Sequence Mining in Categorical Domains: Algorithms and Applications in Sequence Learning: Paradigms, Algorithms, and Applications, Lecture Notes in Computer Science, vol. 1828, February 2001, pp. 162–187.

Vivian F. Lòpez received a Ph.D. in Computer Science from the University of Valladolid in 1996. At present she is Associate Professor at the University of Salamanca (Spain) where she has been since 1998, where at present she is full professor in Computing Science. Member of the Data Mining Group (http://mida.usal.es/) at the University of Salamanca (Spain). She has done research on natural language processing and neural networks. She has also 80 papers published in recognized journals, workshops and conference proceedings, 20 books and book chapters and 20 technical reports most of these in this topics. She has been member of the organizing and scientific committee of several international symposiums. Director of the Master in Intelligent System and Program of Ph.D. in Computer Science at University of Salamanca (Spain).

Ramiro Aguilar Quispe received a Ph.D. in Computer Science at University of Salamanca (Spain) in 2007. He obtained Engineering in Computer Sciences degree in 1999 at the University of San Andrès (Bolivia) and at this moment he is a consultant in system development in La Paz (Bolivia). He has been co-author of published papers in several journals.



Luis Alonso Romero received a M.Sc. in Computing and Control (1970) from the Imperial College of London, and a Ph.D. in Physics (1972) from the University of Valladolid. He was Data Processing Manager at Astano Shipyards (1974–1975), Head of Computing Science Department of University of Valladolid (1992–1996), and Head of Computing Science Department of the University of Salamanca (2000–2004) where at present he is full professor in Computing Science. He was leader of several Artificial Intelligence Projects sponsored by Spanish (CICIT) and European (ESPRIT BR) governments. He is coauthor of more than 120 papers and communications, and supervisor of 11 Ph.D. thesis.

María N. Moreno is Associate Professor at the University of Salamanca, Spain. She is the head of the Data Mining Research Group (http://mida.usal.es/) at the University of Salamanca. She has published more than 100 papers in international journals and conference proceedings. She is co-author of book chapters and has co-edited national and international books. She is reviewer of several journal indexed by ISI Journal Citation Reports and she has been member of the scientific committee of numerous international conferences. Her research interests are in the areas of associative classifiers, multiclassifiers web mining and the application of data mining methods in several domains.

Juan M. Corchado received a Ph.D. in Computer Science from the University of Salamanca in 1998 and a Ph.D. in Artificial Intelligence (AI) from the University of Paisley, Glasgow (UK) in 2000. At present he is Dean at the Faculty of Computer Sciences, Associate Professor, Director of the Intelligent Information System Group (http://bisite.usal.es) and Director of the M.Sc. programs in Computer Science at the University of Salamanca (Spain), previously he was sub-director of the Computer Science School at the University of Vigo (Spain, 1999–2000) and Researcher at the University of Paisley (UK, 1995–1998). He has been a research collaborator with the Plymouth Marine Laboratory (UK) since 1993. He has leaded several Artificial Intelligence research projects sponsored by Spanish and European public and private institutions and has supervised seven Ph.D. students. He is the co-author of over 130 books, book chapters, journal papers, technical reports, etc. published by organizations such as Elsevier, IEEE, IEE, ACM, AAAI, Springer Verlag, Morgan Kaufmann, etc., most of these present.