On Generating and Simplifying Decision Trees Using Tree Automata Models

TALEB ZOUGGAR SOUAD[1]
ATMANI BAGHDAD[2]
ADLA ABDELKADER[3]

Laboratory of Informatics of Oran (LIO), Oran University
LP 1524, El Me Naouer, Es Senia, 31 000 Oran, Algeria.
[1]souad.taleb@gmail.com
[2]atmani.baghdad@univ-oran.dz
[3]adla.abdelkader@univ-oran.dz

Abstract. Tree automata are widely used in applications such as XML document manipulation, natural language processing, and formal verification. We propose in this paper to generate decision trees classifiers using tree automata models. Mainly, two objectives are : 1) fitting these methods in a formal frame and, 2) using tree automata in modeling decision trees and classifying heterogeneous information sources. Specifically, the size of a decision tree is reduced by a post-pruning procedure: a given decision tree is converted into a tree automaton and then the latter is simplified by eliminating useless states and non-determinism. We report some empirical experiments on real-world datasets.

Keywords: Knowledge Discovery, Machine Learning, Classification, Decision Trees, Tree Automata, Determinization.

(Received November 9th, 2013 / Accepted February 6th, 2014)

1 Introduction

In 2011, we started the e-PEV [1]. project which aimed to design and develop a vaccination decision support system. The matter is to realize a complete system implementing the mechanisms of integrating heterogeneous semi-structured data such as XML or natural language in a medical data warehouse and extracting knowledge on which simulations guided by outcomes can be launched to generate and validate the expected assessments. One of the most important applications of XML is data integration from multiple sources which contain diverse information and their presentation to the outside world a schema of the available data.

Tree automata are widely used in applications such as XML document manipulation which consists in a process including incremental validation following document updates, information extraction, wrappers conception, etc. Also, they have been used in pattern recognition tasks to outline some of the item features to be classified. In the e-PEV project, my colleagues contributed to match XML documents and data warehouse [1].

Tree automata are fairly simple computational models. They are common in almost every area of computer science. There are a considerable number of applications in which finite automata are incorporated. One of these applications is natural language processing in which finite automata are used to describe different phases of lexical analysis. Another related application is text processing where automata are used in text compression and file manipulation. Automata are also useful in the area of formal verification and model checking where they can be used to model the systems behavior and to ensure that they work correctly. Recently, we

[1]This work is part of the national research project entitled "Service Oriented Architecture for the widdened vaccination program".

have witnessed the introduction of the XML document processing field. The issue approached by [1] concerns dynamic integration of complex data using cellular automata [4]. The successful use of tree automata in the XML field or in the tree structured data in general spurs us to attempt to make use of their properties, formality and expressive power in the task of XML documents classification [1].

In this paper, we propose a formal framework for decision tree methods [36] and induction graphs [35]. This framework is represented by tree automata which are computational models in finite states that generalize the traditional automata through term or tree recognition. The purpose of this work is to contribute to the use of tree automata simplifying algorithms or properties to post-prune the generated models. The use of automata for decision trees generation [36][35] enables us to propose two simplification algorithms: (1) the first one is based on the cleaning property. This algorithm allows excluding the null rules encountered with the greedy methods by eliminating the non accessible states, which simplifies the rules base by reducing the error rate in some cases. (2) The second algorithm is based on the determinization property in order to reduce the states and transition rules numbers with the aim of reducing the storage complexity. The proposed determinization algorithm is computed in a polynomial time without the use of the subset construction which may cause an exponential blowup of the state number. We show the steps of non-deterministic tree automata generation from a diabetic monitoring decision tree and the classification task carried out through rewriting.

From now on, the paper is organized as follows. In section 2 we give some preliminaries where a review of the relevant research streams is provided: this section is devoted to a presentation of tree automata, machine learning and decision trees. Some related works are outlined in section 3. In section 4, we present our proposal of a tree automaton model for classification. We also present some implementation issues and report some empirical experiments on real-world datasets in section 5. Finally in section 6, the conclusion summarizes the contributions of this work and outlines potential research opportunities in the realm of tree automata and decision trees.

2 Preliminaries

2.1 Tree Automata

Tree automata are computational models which appear in many areas of computer science and engineering [18][17]. They were first used in circuits checking and then in the abstracted interpretation starting from constraints in the context of rewriting problems, automatic theorems evidence or programs checking, etc. They are used in various application fields such as grammatical inference which is a subfield of machine learning [27].

2.1.1 Definitions

"According to [17], A is a finite tree automaton, then A is quadruplet $A = (Q, V, \Delta, Q_f)$where:

- V : Alphabet which elements are functional symbols, a symbol f of arity 1 is noted *f()*, and if it is of arity 2 it is noted *f(,)*, etc,
- Q : Finite set of states,
- Δ : Set of transition rules, the function δ associates a state with a functional symbol,
- $Q_f \subseteq Q$: Set of final or acceptance states".

- For ascending automata, the transition rules are of the form: $\delta(f, q_1, ..., q_s) = q$ for a symbol $f \in V$ with $(arity = s > 0)$. These rules can also be written : $f(q_1, q_2, ..., q_s) \rightarrow q$. Graphically, a transition rule is represented in by the diagram as depicted in Figure 1:

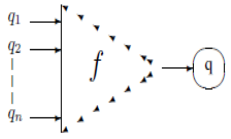

Figure 1: Graphical representation of a transition

- A tree *t* is accepted by the ascending tree automaton A if $\exists q \in Q_f$, $\delta'(t) \rightarrow q, \delta'$ corresponds to successive applications of δ.

- An $\varepsilon - rule$ is a special kind of rules connecting two state, it is of the form : $q_i \rightarrow q_j, q_i, q_j \in Q$.

2.1.2 Deterministic Tree Automata

A tree automaton A= (Q,V,Δ, Q_f) is deterministic, if the relation δ admits at most one image for each state, i.e. if for any sate $q_i \in Q$ and for any symbol $f \in V$such as $arity(f) = n$, the set $\{q|(f, q_1, ..., q_n) = q\}$ is either empty or reduced to a singleton [17].

2.1.3 Cleaning a Tree Automaton

If a state is not accessible, that means that it does not take part in the recognition of any term (tree), it is thus useless. Moreover, any transition driving to it can there be removed.

The automaton size can be significantly reduced by removing from Q any inaccessible state and recursively removing from Δ each transition $f(q_1, ..., q_n) \rightarrow q$ such as q is not accessible [19].

2.2 Machine Learning

Machine learning is based on concepts from the fields of artificial intelligence, statistics, information theory, etc. Information extraction addresses the problem of extracting specific information from a collection of documents. Most works achieving extraction on structured documents use string based matching techniques. The most common type of machine learning is inductive machine learning, where experience is given in the form of learning examples.

We call Ω the training set which allows generating a model φ used to study new examples (prediction) in the same area. Each instance ω of this set is represented by a vector of attributes and each attribute takes a set of values. If the class of each vector is given, it is within the framework of supervised learning whose methods are decision trees [30][31], Bayesian networks, etc.

The generated model, called prediction model or classifier, allows predicting a variable Y, called endogenous variable, class or variable to be explained from a *certain* number of explanatory predictive variables denoted X. The quality of the prediction model or the classification function φ depends on the discrimination capacity which it acquired during the training.

The profile of any instance ω in Ω is described by k explicative or exogenous features X_1, ...,X_k. Those features may be qualitative or quantitative. The variable Y to predict associates, to each individual ω, a class belonging to a set of classes $C = c_1, ..., c_m$. The values taken by the variable Y within the population are discrete − we notice that we deal with symbolic classification. Among the supervised learning methods, we are concerned more particularly in decision trees methods.

2.3 Decision Tree Methods

Data mining has emerged as an important research area of practical interest. It is applied in various fields such as data streams, data-warehouses, and bioinformatics. Decision trees are data mining techniques. They are more attractive than statistical approaches or neural networks, thanks to the interpretability of the provided results. This property is much appreciated because it is absolutely necessary in some real-life applications and pertains to qualified intelligent behavior patterns similar to those in a human being. However, the explanatory power of these methods decreases considerably once the generated trees become big-sized.

Consequently, tree size reduction, also called pruning, becomes indispensable for good interpretability. "In [14], decision trees simplifying methods are categorized across five classes". We are more concerned in the first one and especially in the post-pruning methods which consists in modifying the tree once it has reached complete development [13].

Decision trees are structures used to solve a problem by performing successive tests. Each node corresponds to a test on a discriminating variable of the problem. In general, there is a test in each internal node of the tree which corresponds to an attribute in the training set, and a branch for each of its possible values. At each leaf node, there is a class value. A root to node path is a set of attributes with their values. The target part contains a single attribute, the class. If the latter is nominal, a classification task is required, else if it is numerical, a regression task is considered. These methods generate decision rules in the form of If condition then Conclusion. Splitting criteria are often based on entropies. Among the well-known methods, we cite ID3 [30], C4.5 [31][32], CART [13], etc.

3 Related Work

In the context of the e-PEV project, many researchers strive to develop a platform whose purpose is to bring together various and complementary paradigms, rarely shared by a same project team, such simulations by EASYDEVS [39], NEURO-IG [5] for dimension reductions, artificial learning using CASI [4], etc.

"In [20], the authors studied the training problem of statistical data distribution in a relational database by proposing a method considering the database structure and not requiring any data transformation which may cause a loss of essential information". "In [21], the impact of noise on probabilistic models is examined but the authors didn't propose any functional method for correcting erroneous data". The inference algorithms are based on a merging states approach [15]. The learning process is then performed by merging the states considered statistically equivalent. Habrard and his colleagues [22] proposed a technique for data reduction to eliminate irrelevant or noisy instances from a set of trees. Thomo, and Venkatesh [37] tried to solve the problem by first determining relevant sources and then combining them to produce data which satisfy a given target schema. They used Visibly Pushdown Automata (VPAs) as schema representation for XML documents.

In the same context, many works attempt to use supervised classification of XML documents. "In [25], the

authors proposed a system called "WebClass" to classify web documents using a threefold modified decision tree splitting the root, the depth-one nodes, and the depth-two nodes on keywords, descriptions, and hyperlinks". "In [7], a classification approach of XML documents uses instance-based learning idea". The approach explores documents structure for classification and provides extensions dealing with documents content.

Tree automata have been successfully applied in the context of DTDs (Document Type Definition), the simplest standard for defining the validity of XML documents. The well-known system is XDuce [24], a typed functional language with extended pattern matching operators to manipulate XML documents. In XDuce tool, XML documents types are modeled by regular tree automata while the expressions pattern matching typing is based on closure operations on automata.

Kosala and his colleagues [26] explored methods based on the tree structure of documents. In particular, the method infers a k-testable tree automaton from a small set of annotated examples and proposed different ways to generalize the inferred automaton. Dal Zilio and Lugiez [40] proposed a new class of automata: the sheaves automata and an associated logic dedicated to the interrogation of XML documents. Tree automata are also used in checking document keys constraints following an update [9]. "In another work [8], Bouchou and Alves used automata to perform incremental testing by checking only the part of the document concerned with updates, the validity of a document". Chidlovskii [16] used regular tree automata to model XML documents and showed that tree automata are more powerful than the XML DTDs. The wrapping is achieved considering query-algebra-based tree automata models and queries optimization techniques. The authors also showed the conversion of tree automata schema into XML DTDs. The wrapping is achieved considering query-algebra-based tree automata models and queries optimization techniques. The authors also showed the conversion of tree automata schema into XML DTDs.

Automata minimization has been extensively studied and it has been found that minimal automaton constitutes a fundamental principle of approaches using and implementing finite automata tools in areas such as word processing, image analysis, computational linguistics, and many other applications [6]. There are two main classes of minimization algorithms: The first performs by successive refinements of a set of states, and the second operates by states fusion [33][2]. Among the first class, Brainerd [10] proposed an algorithm in which the minimal tree automaton is computed by constructing congruencies of the input automaton until a fixed point is reached. An effective algorithm (in terms of time complexity) for the minimization of accessible and complete deterministic automata was proposed by Hopcroft [23]. To minimize a non-deterministic finite automaton, a corresponding deterministic automaton is computed using subsets construction, which may lead to a combinatorial explosion of the automaton size. Then, the resulting automaton is minimized. To avoid the subset construction, the size of a non-deterministic finite automaton is released using heuristic methods, for example, by identifying and eliminating states that are equivalent according to some equivalence relation [29][28].

Finite tree automata are natural generalizations of word automata. Whereas a word automaton accepts a single word, a tree automaton accepts a tree (term). The fact that both word and tree automata are used in so many different areas of computer science has generated a growing number of software systems designed where automata are used as internal representations.

Sempere and Lopez [34] proposed a method which combines decision trees and tree automata to solve a pattern recognition problem simplified here to digits recognition. Their resolution was carried out in two stages: grammatical inference is used to construct a tree automaton from every set of trees taken in a quad tree representing the same digit. A learning stage allowed obtaining a different representation of a digit based on distances of every digit (every tree) to every concept (every automaton). A decision tree is inferred by using standard methods based on the entropy of the examples and distances.

4 A Proposal of a Tree Automaton Model for the Classification

4.1 Our Approach

The simplification of a decision tree generated by a tree automaton (see Figure 2) is realized in different steps. First, data is preprocessed by eliminating redundancy and dealing with the missing values, then we proceed to the tree automaton generation using decision tree computing. The generated automaton can be visualized graphically or in a text form, and the extraction of decision rules is launched using a rewriting process [35]. Finally, the original automaton is simplified and a validation phase is carried out directly over the tree automaton to ensure that the simplification operation don't entail performance degradation.

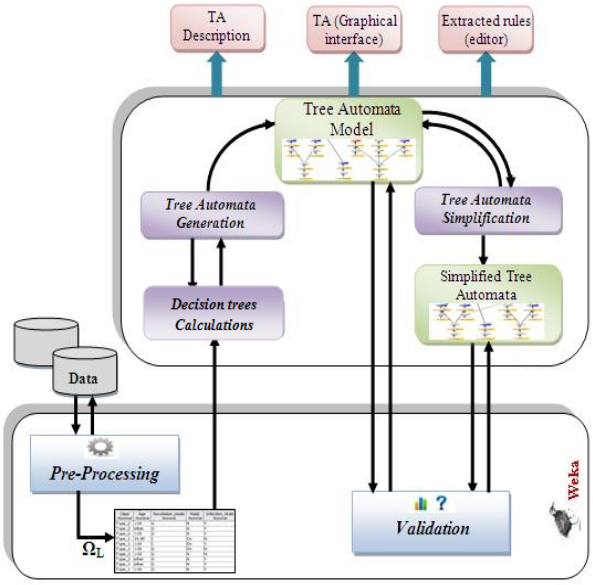

Figure 2: Steps of generation and simplification of a decision tree by a tree automaton

4.2 Illustration Through an Example

The example of table 1 consists of 20 instances or individuals described by 04 exogenous variables: Day, Season, Wind and Rain in order to determine if an individual comes On Time (T), Late (L), Very Late (VL), Do not come or Cancelled (C). These indications represent the modalities of the class or the endogenous variable.

Table 1: Training sample [11]

Day	*Season*	*Wind*	*Rain*	*Class*
Weekday	Spring	None	None	T
Weekday	Winter	None	Slight	T
Weekday	Spring	None	Slight	T
Weekday	Winter	High	Heavy	L
Saturday	Summer	Normal	None	T
Weekday	Autumn	Normal	None	VL
Holiday	Summer	High	Slight	T
Sunday	Summer	Normal	None	T
Weekday	Winter	High	Heavy	VL
Weekday	Summer	None	Slight	T
Weekday	Summer	High	Slight	T
Saturday	Winter	Normal	None	L
Weekday	Summer	High	None	T
Weekday	Winter	Normal	Heavy	VL
Saturday	Autumn	High	Slight	T
Weekday	Autumn	None	Heavy	T
Holiday	Spring	Normal	Slight	T
Weekday	Spring	Normal	None	T
Weekday	Spring	Normal	Slight	T
Saturday	Spring	High	Heavy	C

The decision tree generated in the Weka platform [38] for the sample of table 1 is presented in Figure 3.

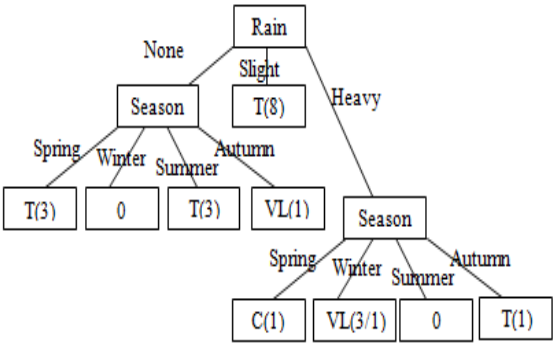

Figure 3: C4.5 decision tree of the sample presented in Table 1

From the generated decision tree, we obtain the following rules:

R1- If Rain = 'None' And Season= 'Spring' Then Class= 'T' (100%)

R2- If Rain = 'None' And Season= 'Winter' Then Class= 'null' (0%)

R3- If Rain = 'None' And Season= 'Summer' Then Class= 'T' (100%)

R4- If Rain = 'None' And Season= 'Autumn' Then Class= 'VL' (100%)

R5- If Rain = 'Slight' Then Class= 'T' (100%)

R6- If Rain = 'Heavy' And Season= 'Spring' Then Class= 'C' (100%)

R7- If Rain = 'Heavy' And Season= 'Winter' Then Class= 'VL' (75%)

R8- If Rain = 'Heavy' And Season= 'Summer' Then Class= 'null' (0%)

R9- If Rain = 'Heavy' And Season= 'Autumn' Then Class= 'T' (100%)

Among the nine extracted rules, the rules R2 and R8 are null. These are rules for which the value of the class is unknown owing to the fact that the associated manpower of the leaf is null.

Considering the tree automata formalism presented in [36], while basing on decision trees transformation operations, the tree automaton represented on Figure 4 and corresponding to the decision tree of Figure 3 is constituted of the following elements:

- Q_f= $\{q_0, q_1, q_2, q_3\}$, each of these states is related to a modality of the class,

- V= {Season (), Spring, Summer, Autumn, Winter, Day (,), Weekday, Saturday, Sunday, Holiday, Wind (,), High, Normal, None, Rain (,), Heavy, Slight, None},

- Q = $\{q_0, q_1, ..., q_{25}\}$, $|Q|$= 26,

- Δ= {T $\rightarrow q_0$, L$\rightarrow q_1$, VL $\rightarrow q_2$, C $\rightarrow q_3$, None $\rightarrow q_4$, Heavy $\rightarrow q_6$, Slight $\rightarrow q_5$, Rain $(q_5) \rightarrow q_8$, Rain $(q_4) \rightarrow q_7$, Rain $(q_6) \rightarrow q_9$, Summer $\rightarrow q_{12}$, Summer $\rightarrow q_{20}$, Winter $\rightarrow q_{13}$, Winter $\rightarrow q_{21}$, Spring $\rightarrow q_{10}$, Spring $\rightarrow q_{19}$, Autumn $\rightarrow q_{11}$, Autumn $\rightarrow q_{18}$, Season $(q_7, q_{12}) \rightarrow q_{16}$, Season $(q_7, q_{13}) \rightarrow q_{17}$, Season $(q_7, q_{10}) \rightarrow q_{14}$, Season $(q_7, q_{11}) \rightarrow q_{15}$, Season $(q_9, q_{20}) \rightarrow q_{24}$, Season $(q_9, q_{21}) \rightarrow q_{25}$, Season $(q_9, q_{18}) \rightarrow q_{22}$, Season $(q_9, q_{19}) \rightarrow q_{23}$,$q_8 \rightarrow q_0$, $q_{16} \rightarrow q_0$, $q_{14} \rightarrow q_0$, $q_{22} \rightarrow q_0$, $q_{15} \rightarrow q_2$, $q_{25} \rightarrow q_2$, $q_{23} \rightarrow q_3$}. $|\Delta|$ = 33.

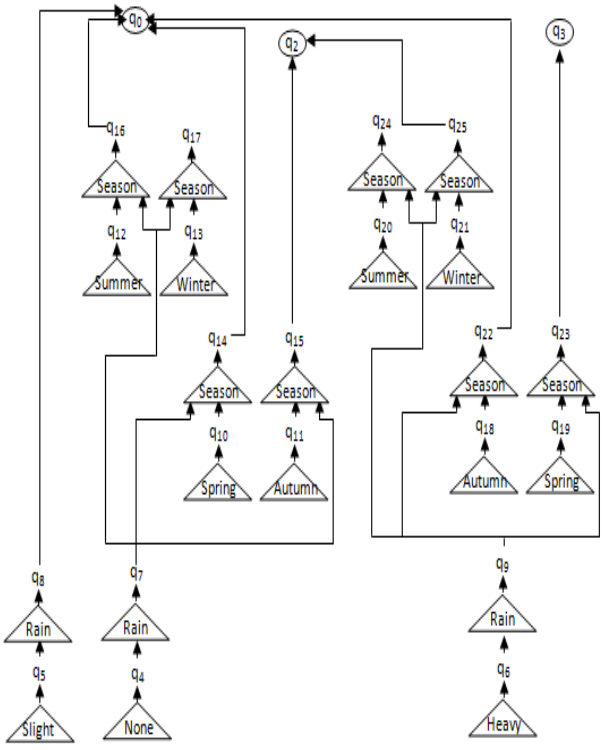

Figure 4: The tree automaton corresponding to the decion tree of Figure 3

The extraction of decision rules from the automaton is done by rewriting as presented in [35], this allows us to obtain the same rules extracted from the tree presented in Figure 3.

4.3 Cleaning a Tree Automaton

According to the definition of cleaning, a state is useless if it does not participate to the recognition of any term (tree). In the classification case, a term is a decision rule.

A state is accessible if a final state is reached from it (expressing a decision) either in a direct or indirect way.

We present below the proposed automaton cleaning algorithm:

ALGORITHM CLEANING,
Require: Automaton $A = (Q, V, \Delta, Q_f)$
US = $|Q - Q_f|$,
1:For i= 1 To US **Do**
2:While j $\leqslant |\Delta|$ **Do**
3:If $q_i \in$ Source (rule$_j$) Then
. US $\leftarrow$ US - q_i,
4:End While,
5:End For,
6:While US $\neq \varnothing$ **Do**
7:While j $\leqslant |\Delta|$ **Do**
8:If q_i = Target (rule$_j$) Then
. Delete rule$_j$ from Δ,
. US$\leftarrow$ US - q_i,
. $Q \leftarrow Q - q_i$,
. $V \leftarrow V$ - Symbol(rule$_j$),
9:End While,
10:End For,
US = $|Q - Q_f|$,
goto 1,
Return A Cleaned Automaton $A = (Q, V, \Delta, Q_f)$.

The algorithm can be used for any automaton and returns a cleaned one. The algorithm consists of two main parts:

(1) It tries first to find the US, initially US contains all states of Q except final states, set of states that appear in the source of at least one transition rule,

(2) In the second part (two last loops), the sets Q, and US are updated by eliminating useless states. Consequently symbols and rules containing these useless states are eliminated V, Δ.

As illustration, the application of the cleaning algorithm to the example presented in section 4.2 allows modifying sets Q and Δ as follow:

For the two first loops (corresponding the part one) we build the set US, this set is only constituted of states which are sources of at least one transition rule: US=$\{q_{17}, q_{24}\}$.

The two last loops (part two) allow updating the sets US, Δ, and Q:

US=$\{q_{17}, q_{24}, q_{13}, q_{20}\}$,

Δ = Δ-{Season $(q_7, q_{13}) \rightarrow q_{17}$, Winter$\rightarrow q_{13}$, Season $(q_9, q_{20}) \rightarrow q_{24}$, Summer$\rightarrow q_{20}$},

The sets sizes are reduced from $|Q|$=26, $|\Delta|$=33 to $|Q|$=24,$|\Delta|$=29 once the cleaning is achieved. Furthermore, the simplification of the automaton by eliminating the ε rules [35] allows reducing even more states and transition rules numbers $|Q|$=17 and $|\Delta|$=22.

4.4 Elimination of Non-Determinism

An automaton is deterministic if there do not exist two rules with identical sources (left sides). We notice that the tree automaton of the example is nondeterministic, the subset of rules which causes the non-determinism is {Spring→ q_{10}, Spring→ q_{19}, Autumn→ q_{11}, Autumn→ q_{18}, Summer→ q_{12}, Summer→ q_{20}, Winter→ q_{13}, Winter→ q_{21}}.

Another algorithm presented "in [17] makes deterministic an automaton which is not but contrary to our determinization approach this algorithm increases considerably the state number of the deterministic automaton".

It is noticed that non-determinisms are caused by rules whose source are labeled by a symbol of arity 0. The transcribing steps force the arity of any symbol to be maximum equal to 2 and minimum to 0. The symbols of arity 1 or 2 can't cause non-determinism because they differ by their arguments but the arity 0 symbols have no arguments so when they reappear as source of more than one rule they are always the same and consequently cause non-determinism.

This makes the task of determinization simpler and for which we can present the following generic algorithm:

ALGORITHM DETR,
Require: Non deterministic Cleaned Automaton $A = (Q, V, \Delta, Q_f)$,
1: For i=1 To |Δ| **Do**
2: For j=2 To |Δ| **Do**
3: If Source (rule$_i$)=Source (rule$_j$) Then
. $Q = Q-${Target(rule$_j$)};
. Replace in Δ target(rule$_j$) by target(rule$_i$);
. Δ=Δ-{rule$_j$};
4: End For;
5: End For;
Return Deterministic Automaton $A = (Q, V, \Delta, Q_f)$.

The algorithm operates on any automaton and returns a deterministic one, in this case we assume that the automaton is cleaned because we apply the algorithms in order: cleaning, and then determinization.

DETR is composed of two loops in which we try to detect rules causing non-determinism, leaving just one of them in the set Δ. The application of the algorithm to the example allows obtaining a deterministic automaton with reduced sets of transition rules and states.

Δ={T → q_0, L→ q_1, VL → q_2, C → q_3, None → q_4, Heavy → q_6, Slight → q_5, Rain (q_5) → q_0, Rain (q_4) → q_7, Rain (q_6) → q_9, Summer → q_{12}, Winter → q_{21}, Spring → q_{10}, Autumn → q_{11}, Season (q_7, q_{12}) → q_0, Season (q_7, q_{10}) → q_0, Season (q_7, q_{11}) → q_2, Season (q_9, q_{21}) → q_2, Season (q_9, q_{11}) → q_0, Season (q_9, q_{10}) → q_3}.

With the non-determinism elimination, we obtain the sets Q and Δ for which respective cardinalities are 15 and 20. The non-determinism is due primarily to the reappearance of the variables already used in the tree in other positions.

5 Experimental Results

5.1 Presentation of the Application Domain

We are interested to the domain of diabetic patients monitoring. The diabetes monitoring allows verifying if a diabetic patient presents complications. The development of the application MONITDIAB was accomplished based on several doctors' interviews and consultation of patients' archives.

We aimed to collect all the data allowing to define if a patient presents any complication, the doctors opinion served us to reveal the symptoms leading to complications, then the archives consultation allowed us to gather the real cases already processed in the hospital. Besides, we have attended many consultations.

To elaborate the classification model, as explained in section 4, we need exogenous variables which we extract from the patients checks up. The class corresponding to complications can be: Unbalanced Diabetes Slightly Complicated (UDSC); Unbalanced Diabetes Moderately Complicated (UDMC); Unbalanced Diabetes Very Complicated (UDVC); Unbalanced Diabetes Non Complicated (UDNC); Balanced Diabetes Non Complicated (BDNC).

13 exogenous variables have been extracted:
1- Diabetes Type : (DT)
- Type I : Insulin-Dependent Diabetes (IDD) Is characterized by sometimes a complete disappearance of the secretion of insulin by the pancreas, having for immediate consequence a hyperglycaemia, the treatment by insulin allows the patient to live an almost normal life.
- Type II : Non Insulin-Dependent Diabetes (NIDD) The most frequent type of diabetes, it represents a heterogeneous infection in the pathogenic, clinical and biological triple plan.
2- Var : Determines cases where the patient is :
- A Pregnant Female (PF);
- An adult (age <= 70) (Adult);
- An old person (age > 70) (OP);
3- BMI : (Body Mass Index): BMI = $Weight/(Height)^2$
- 18<BMI<20 => Meagre or Thin (M);
- 20<BMI<25 => Normal;
- 25<BMI<30 => Excess Weight (EW);

- 30<BMI<35 => Obesity G1;
- 35<BMI<40 => Obesity G2;
- BMI>40 => Obesity G3;

4- The glycaemia:(Glyc) The glycaemia depends on the sex and on the age of the patient and its value is in balance if it is:
- Equal to 1 G (on an empty stomach) and 1.20 G (in post prondial) for the pregnant woman,
- Equal to 1.60 G (on an empty stomach) and 2 G (in post prondial) for an old person,
- Equal to 1 G (on an empty stomach) and 1.40 G (in post prondial) for a young or an adult subject.

The values taken by this exogenous variable are:
- If 0.70 G<=Glyc< 1.80 G => Normal;
- If Glyc < 0.70 G =>Hypoglycaemia (HypoG);
- If 1.80 G<=Glyc<= 6 G => Hyperglycaemia (HyperG);

5- The HBANC: (HBANC) reflects the glycaemic balance from three months:
- 6.5 < HBANC < 7 Balanced for a Type I Diabetes (B);
- 7 < HBANC < 7.5 Balanced for a Type II Diabetes (B);
- 8 < HBANC < 10 Unbalanced (U);
- HBANC > 10 Very Unbalanced (VU);

6- Eye bottom examination: (EyeEx)
- No Retinopathy (NR),
- Retinopathy (R),

7- The creatinine: (Crea)
- 6G / L <= Crea <= 13 G/L it is Normal,
- Crea > 13 G/L it is Anormal,

8- Urea: (Urea)
- 0.30 G/L <= Urea <= 0.50 G/L => Normal,
- Urea > 0.50 G/L => Anormal (Renal Insufficiency),

9- Microalbuminuria: (McrAlb)
- McrAlb = 20mg /24h => Normal,
- 30 < McrAlb < 100=> Diabetic Nephropathy stage 3A (DNS3A),
- 100 < McrAlb < 300=> Diabetic Nephropathy stage 3B (DNS3B),
- McrAlb > 300=> Diabetic Nephropathy stage 4 (DNS4),
- McrAlb > 300 and high Urea and high crea => stage 5 (Renal Insufficiency) (RIS5),

10- Clearance of creatinine : (Cc)
The value of Cc is calculated using the formula Cc= (140 - (age*weight))/creatinine and can take the following values:
- Cc > 100 Normal;
- 70 < Cc < 100 Slight Renal Insufficiency (SRI);
- 40 < Cc < 70 Moderate Renal Insufficiency (MRI);
- 10 < Cc < 30 Severe Renal Insufficiency (SVRI);
- Cc < 10 Very Severe Renal Insufficiency (VSVRI);

11- Neuropathy : Can appear after five years of diabetes discovery:
- Neuropathy existance (Neurpath);
- No Neuropathy (No_Neurpath);

12- ECG :
- Normal ;
- Coronary Insufficiency (CorInsuf);
- Cardiac Insufficiency (CarInsuf);

13- Arterial Doppler: (AD)
- Arteropathy Existance (Art);
- No Arteropathy (NoArt);

The MONITDIAB application processed 353 patients and 13 exogenous variables, an Excerpt of the application is presented on Figure 5.

DT Nominal	Var Nominal	BMI Nominal	Glyc Nominal	HBANC Nominal	Exeye Nominal	Crea Nominal	Urea Nominal
TypeI	PF	EW	HyperG	U	R	Normal	Normal
TypeI	Adult	EW	HyperG	U	R	Normal	Normal
TypeI	Adult	M	HyperG	U	R	Normal	Normal
TypeI	Adult	M	HyperG	U	R	Normal	Normal
TypeI	Adult	Obesit...	HyperG	U	R	Normal	Normal
TypeII	Adult	Obesit...	HyperG	U	R	Normal	Normal
TypeII	Adult	Obesit...	HyperG	U	R	Normal	Normal
TypeII	Adult	Obesit...	HyperG	U	R	Normal	Normal
TypeII	Adult	Obesit...	HyperG	U	R	Normal	Normal
TypeII	Adult	Obesit...	HyperG	U	R	Normal	Normal

Figure 5: Excerpt of the MONITDIAB base

5.2 The Execution Process

The tree automaton carrying out the classification task, on the MONITDIAB application by using the principles of the C4.5 [32] method, is presented in two different ways using a graphical representation in Figure 6 or by the presentation of the sets Δ, Q and V in a textual form as shown in Figure 7.

The respective cardinalities of the sets Δ , Q, and V are: 75, 63, and 21.

The number of decision rules extracted from the automaton is 19 among which 7 are null rules, the rules are presented on Figure 8.

The validation is carried out using cross validation 10-folds available on the platform. The classification performances of the rough automaton applied to the MONITDIAB base are presented in Figure 9:

The application of the simplification algorithms, previously presented allows us to obtain the simplified automaton which the sets Δ , Q, and V have respectively the cardinalities 37, 37, and 17. The decision rules number extracted from the simplified automaton is 12.

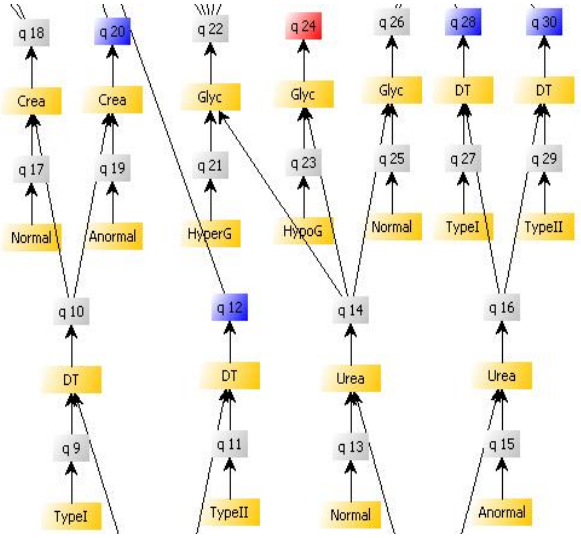

Figure 6: Tree automaton generated for the MONITDIAB application

```
V = { R ,Exeye(,) ,NR ,TypeI ,DT(,) ,TypeII ,
      Normal ,Urea(,) ,Anormal ,Crea(,) ,HyperG ,Glyc(,) ,
      HypoG ,Neurpath ,Neuropathy(,) ,No_Neurpath ,Cc(,) ,SRI ,
      SVRI ,MRI ,VSVRI }

Q = { q0 ,q1 ,q2 ,q3 ,q4 ,q5 ,q6 ,q7 ,q8 ,q9 ,q10 ,q11 ,q12 ,q13 ,q14
      q20 ,q21 ,q22 ,q23 ,q24 ,q25 ,q26 ,q27 ,q28 ,q29 ,q30 ,q31 ,q32
      q40 ,q41 ,q42 ,q43 ,q44 ,q45 ,q46 ,q47 ,q48 ,q49 ,q50 ,q51 ,q52
      q60 ,q61 ,q62 }

Delta = { R(q4)--> q5, Exeye(q5)--> q6, NR(q6)--> q7,
          Exeye(q7)--> q8, TypeI--> q9, TypeII(q10,q6)--> q11, q12--> q0,
          Normal(q12,q6)--> q13, Anormal(q14,q8)--> q15, Normal(q16,q8)--
          Anormal(q18,q10)--> q19, q20--> q0, HyperG(q20,q10)--> q21,
          HypoG(q22,q14)--> q23, Normal(q24,q14)--> q25, TypeI(q26,q14)--
          q28--> q1, TypeII(q28,q16)--> q29, q30--> q0, Neurpath(q30,q16)
          No_Neurpath(q32,q18)--> q33, q34--> q4, Normal(q34,q18)--> q35,
```

Figure 7: The sets Δ , Q and $V$of the rough automaton for the MONITDIAB base

```
- If DT = TypeII And Exeye = R Then Complications = UDVC .
- If Crea = Anormal And DT = TypeI And Exeye = R Then Complications = UDVC .
- If Glyc = HypoG And Urea = Normal And Exeye = NR Then Complications = null .
- If DT = TypeI And Urea = Anormal And Exeye = NR Then Complications = UDMC .
- If DT = TypeII And Urea = Anormal And Exeye = NR Then Complications = UDVC .
```

Figure 8: Excerpt of the rules base generated from the rough automaton

```
Correctly Classified Instances          344               97.4504 %
Incorrectly Classified Instances          9                2.5496 %
Kappa statistic                           0.9528
Mean absolute error                       0.0128
Root mean squared error                   0.099
Relative absolute error                   5.8055 %
Root relative squared error              29.9468 %
Total Number of Instances               353
```

Figure 9: Classification results for the rough automaton

5.3 Comparative Study

The comparative study depicted in table 2 is carried out between a rough automaton generated by using C4.5 [32] method, and the simplified automaton obtained from the rough automaton by applying determinization and cleaning algorithms. We use cross validation 10-folds for the estimation of error rate and datasets from [3] for the comparison purpose.

The comparison is achieved between a rough and a simplified automaton which are represented by the column AT (Automaton Type):
- The rough Automaton (R) : Indicates the initial generated automaton, on which no simplification was brought,
- The simplified Automaton (S) : Indicates the automaton one applying the simplification properties.
For the two automata, we define: SN: States Number, TRN: Transition Rules Number, DRN: Decision Rules Number, and ER: Error Rate.

Table 2: Experimental results

Dataset	AT	SN	TRN	DRN	ER
MONIT-DIAB	R	63	75	19	02.55%
	S	37	37	12	02.55%
Balance-Scale	R	207	259	52	23.34%
	S	110	110	49	23.34%
Breast-Cancer	R	12	16	4	24.58%
	S	12	12	4	24.58%
Car	R	336	497	131	07.63%
	S	196	196	131	07.63%
Credit	R	32	42	11	21.73%
	S	26	26	10	21.21%
Diabetes	R	32	41	11	0.34%
	S	24	24	09	0.34%
Flags	R	144	176	50	40.68%
	S	58	58	25	38.63%
Kr-Ks-Vp	R	118	149	31	0.57%
	S	65	65	31	0.57%
Moonks-Problem1	R	36	47	12	17.88%
	S	20	20	10	17.05%
Solar-flare	R	70	92	23	27.88%
	S	47	47	21	27.52%

We also made a comparison with several benchmarks between C4.5 trees [32] generated and simplified by tree automata and others simplified by a pruning algorithm available on the Weka platform. Table 3 presents the Decision Rules Number (DRN) and the Error Rate (ER) for five benchmarks for which the generated trees are simplified by tree automata called Tree Automata Pruning (TAP) and by C4.5 Pruning algorithm (C4.5 Pruning):

Table 3: Comparative study using TAP and C4.5 Pruning

	TAP		C4.5 Pruning	
	DRN	ER%	DRN	ER%
Autos	30	18.04	49	18.11
Bridges-V1	08	36.36	09	44.00
Lymphography	18	21.71	21	23.04
Mfeat-pixel	296	16.39	708	21.34
Splice	124	05.61	184	05.92

6 Conclusion & Future Work

Finite tree automata are generalizations of words automata structured in trees and they keep their essential properties: they have good logical and ensembles' properties, along with effective algorithmic. These characteristics allow for a large number of implementations, especially as far as typing and programs analysis are concerned. Also, they allow defining regular trees languages with an operational semantic, applying a general frame for XML languages and supplying a validation execution environment while serving as basic tool for the static analysis (proofs, decision-making procedures in logic).

We use tree automata models for the generation of classifiers based on decision trees and induction graphs. Their utilization puts these methods, applied in various and very sensitive fields, in a formal framework.

In addition to the formalization of these methods, the bottom line is to exploit the properties of tree automata and their simplification algorithms in order to simplify the classification models which require precision and are time and space constrained.

By applying determinization and ε-rules elimination properties, we notice that states and transition rule numbers fall considerably without causing performances' degradation. As for the cleaning property, it automatically causes the reduction of the states and transition rules numbers as well as the simplification of the decision tree by eliminating the null rules.

This may simplify the rules base by trimming their number and, in some cases, by reducing the error rate in generalization. The minimization of the generated automata models allows quick classifying and cuts down the response time.

According to the experiments, the proposed approach significantly reduces the decision tree size without hurting effectiveness. Also, it outperformed the pruning algorithm provided by the Weka Project.

As we noted in the previous sections, further work based on using conversion of tree structured data and rewriting techniques to rules base generation for XML documents classification [1] need to be done. The second is to use simplification algorithms to simplify even more the generated tree automaton. Finally, we propose to use tree automata for the generation and the simplification of random forests [12] consisting of C4.5 [32] decision trees.

References

[1] Abdelouhab, F. and Atmani, B. Extracting structure of an xml document : Boolean modeling. *ASD 2009 Workshop on Decisionnel Systems*, pages 67–81, 2009.

[2] Almeida, J. and Zeitoun, M. Description and analysis of a bottom-up dfa minimization algorithm. *Inf. Process. Lett.*, 107:52–59, July 2008.

[3] Asuncion, A. and Newman, D. J. UCI machine learning repository, 2007.

[4] Atmani, B. and Beldjilali, B. Knowledge discovery in database: Induction graph and cellular automaton. *Computing and Informatics*, 26(2):171–197, 2007.

[5] Atmani, B. and Beldjilali, B. Neuro-ig: A hybrid system for selection and elimination of predictor variables and non relevant individuals. *Informatica, Lith. Acad. Sci.*, 18(2):163–186, 2007.

[6] Berstel, J., Boasson, L., Carton, O., and Fagnot, I. Minimization of automata. *CoRR*, abs/1010.5318, 2010.

[7] Bouchachia, A. and Hassler, M. Classification of xml documents. In *CIDM*, pages 390–396, 2007.

[8] Bouchou, B. and Alves, M. H. F. Updates and incremental validation of xml documents. In *DBPL*, pages 216–232, 2003.

[9] Bouchou, B., Alves, M. H. F., and Musicante, M. A. Tree automata to verify xml key constraints. In *WebDB*, pages 37–42, 2003.

[10] Brainerd, W. S. The minimalization of tree automata. *Information and Control*, 13(5):484–491, 1968.

[11] Bramer, M. *Principles of Data Mining*. Springer, 2007.

[12] Breiman, L. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[13] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth, 1984.

[14] Breslow, L. A. and Aha, D. W. Simplifying decision trees: A survey. *The Knowledge Engineering Review*, 12(01):1–40, 1997.

[15] Carrasco, R. C. and Oncina, J. Learning stochastic regular grammars by means of a state merging method. pages 139–152. Springer-Verlag, 1994.

[16] Chidlovskii, B. Using regular tree automata as xml schemas. In *Proc. IEEE Advances on Digital Libraries Conference 2000*, pages 89–98, 1999.

[17] Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[18] Gécseg, F. and Steinby, M. *Tree Automata*. Akadéniai Kiadó, Budapest, Hungary, 1984.

[19] Genet, T. and Gnaedig, I. Termination proofs using gpo ordering constraints. In *TAPSOFT*, pages 249–260, 1997.

[20] Habrard, A., Bernard, M., and Jacquenet, F. Generalized stochastic tree automata for multi-relational data mining. In *ICGI*, pages 120–133, 2002.

[21] Habrard, A., Bernard, M., and Sebban, M. Improvement of the state merging rule on noisy data in probabilistic grammatical inference. In *10th European Conference on Machine Learning. Number 2837 in LNAI, Springer-Verlag*, pages 169–1180, 2003.

[22] Habrard, A., Bernardc, M., and Sebban, M. Detecting irrelevant subtrees to improve probabilistic learning from tree-structured data. *Fundam. Inf.*, 66:103–130, November 2004.

[23] Hopcroft, J. E. An n log n algorithm for minimizing states in a finite automaton. Technical report, Stanford, CA, USA, 1971.

[24] Hosoya, H. and Pierce, B. Regular expression pattern matching for xml. In *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '01, pages 67–80, New York, NY, USA, 2001. ACM.

[25] Hu, W.-C. Webclass: Web document classification using modified decision trees. In *In Proceedings of the 38 th Annual ACM Southeast Conference*, pages 262–263, 1999.

[26] Kosala, R., Blockeel, H., Bruynooghe, M., Van, J., Limburgs, B., and Centrum, U. Information extraction from structured documents using k-testable tree automaton inference, 2006.

[27] la Higuera, C. D. A bibliographical study of grammatical inference. *Pattern Recogn.*, 38:1332–1348, September 2005.

[28] Parosh, A. A., Bouajjani, A., Holik, L., Kaati, L., and Vojnar, T. Composed bisimulation for tree automata.

[29] Parosh, A. A., Bouajjani, A., and Kaati, L. Computing simulations over tree automata: Efficient techniques for reducing tree automata. In *In Proc. of TACAS'08, LNCS*. Springer, 2008.

[30] Quinlan, J. R. Induction of decision trees. *Mach. Learn*, pages 81–106, 1986.

[31] Quinlan, J. R. *Probabilistic decision trees*, pages 140–152. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[32] Quinlan, J. R. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[33] Revuz, D. Minimisation of acyclic deterministic automata in linear time. *Theor. Comput. Sci.*, 92:181–189, January 1992.

[34] Semper, J. and Lopez, D. *Learning decision trees and tree automata for a syntactic pattern recognition task*, volume 2652, pages 943–950. Springer, 2003.

[35] Taleb, Z. S. and Atmani, B. Extraction des règles á partir des données: Graphes d'inductions et automates d'arbres. In *The 3rd Workshop on the Decisional Systems (ASD'08)*, pages 185–196, 2008.

[36] Taleb, Z. S. and Atmani, B. Optimization of a tree automata model for the classification. In *International Arab Conference on Information Technology (ACIT'08)*, 2008.

[37] Thomo, A. and Venkatesh, S. Rewriting of visibly pushdown languages for xml data integration. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 521–530, New York, NY, USA, 2008. ACM.

[38] Witten, I. H. and Eibe, F. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[39] Zeigler, B. P., Praehofer, H., and Kim, T. G. *Theory of Modeling and Simulation, Second Edition*. Academic Press, 2 edition, 2000.

[40] Zilio, S. D. and Lugiez, D. Xml schema, tree logic and sheaves automata. In *In Proc. RTA 2003. LNCS 2706*, pages 246–263. Springer, 2003.