Computing and Informatics, Vol. 26, 2007, 123–147

TOWARDS HYBRIDIZATION OF KNOWLEDGE REPRESENTATION AND MACHINE LEARNING

Ahmed Khorsi

Research Center, Djillali Liabes University
Bel Abbes, 22000, Algeria
e-mail: `ahmed-khorsi@univ-sba.dz`

Manuscript received 29 November 2005; revised 6 February 2007
Communicated by Jacek Kitowski

Abstract. Machine learning and knowledge representation are two fields of artificial intelligence that lead with intelligent reasoning, each one differently. When Knowledge Representation (KR) focuses more on the epistemological face of knowledge to carry out a power-expression model with detriment of the computation efficiency. Machine learning pays more attention to the computation efficiency, often with loss of expressing power. In this paper we show that features of one may overwhelm drawbacks of the other. Taking the uncertainty artefact from machine learning and symbolic representation from KR, we propose in this paper a new memory modelling for knowledge based systems which is at the same time machine-learning structure and a knowledge representation model. In terms of machine learning, our structure allows an unlimited flexibility where no restraining architecture is imposed at the beginning (think about decision trees [18, 6, 5, 2]). Classification can be performed with incomplete vectors where the most likely corresponding class is assigned to the vector with missing attributes. Viewed as a knowledge model, a basic knowledge is easily specified graphically. Inference is defined by rules expressed in the same manner, where the existing sub-instances are used to generate new connections and entities. Inference on existing knowledge is described in two algorithms. Our approach is mainly based on the representation of the *context* concept. Our model brings together advantages of the symbolic knowledge representation, namely human to computer knowledge coding, and those of machine learning structures, namely ease of efficient coding of inference to perform the so-called intelligent tasks like pattern recognition, prediction and others. Its graphical representation allows visualization of both dynamic and static sides of the model (i.e. inference and knowledge).

Keywords: Knowledge representation (KR), machine learning (ML), memory modelling

1 INTRODUCTION

When researchers in KR tempt to simplify knowledge expressing and give less attention to the computation performances [4], those in machine learning [23] tend to design structures that make computation more efficient. Learn automatically means learn from examples. This means that the acquired knowledges are uncertain. Hence, each structure in ML has its own manner to symbolize this uncertainty [20, 17].

Section two is a description of neural nets which shows the main characteristics of machine learning structures and semantic net which is one of the first and most used knowledge models which also may be viewed as a representative model in KR. This will introduce our idea of merging features of KR and ML in order to carry out a knowledge model that allows powerful and flexible knowledge expressing and efficient computation both in learning and inference.

The model proposed in [12] which will be described in Section three is an attempt to endow the knowledge representation with a simple uncertainty modelling. This was the first version of our model. Section four will show how our model represents a simple Boolean function and how its architecture allows easy addition of new knowledge. This is a ML-like approach to show capabilities of our representation. Section five shows how inference rules are defined. New knowledge is intuitively and efficiently added. This may be viewed as a KR-like approach to show expressing capabilities of our model. Section six details the representation of weights which express uncertainty in our model. This representation allows unsupervised learning as well as machine learning structures do.

2 NEURAL NETWORKS AND SEMANTIC NETWORKS

Comparing our model to all others may take hundreds of pages. We restrict ourselves to give representative and well known examples. Neural nets is a well explored and nowadays well exploited machine learning structure, where semantic nets is one of the primary knowledge representation models, but also one of the most used in the modern sub fields of AI (eg. *ontologies*).

2.1 Neural Networks

The pioneers of neural networks [26, 14, 19, 7, 9, 22] were McCuloch and Pitts who designed a simple network using electrical circuits. The neural networks are quite famous by being well adapted for classification problems.

As its name suggests, a neural net is a set of connected logical units that form a network with successive layers. The outputs of each unit are inputs of units (neurons) of the following layer. The inputs of the neurons of the first layer are the components of the characteristic vector, while the outputs of the last layer are the classification results. The layers between the first and the last are called *hidden layers*. The output function of each neuron is in the form:

$$o = \phi\left(\sum_{i=1}^{k} w_i x_i + b\right)$$

where ϕ is a non-linear function such as $\frac{1}{1+e^{ax}}$ or $\tanh(x)$. Figure 1 shows a graphical representation of a multi-layer neural network.

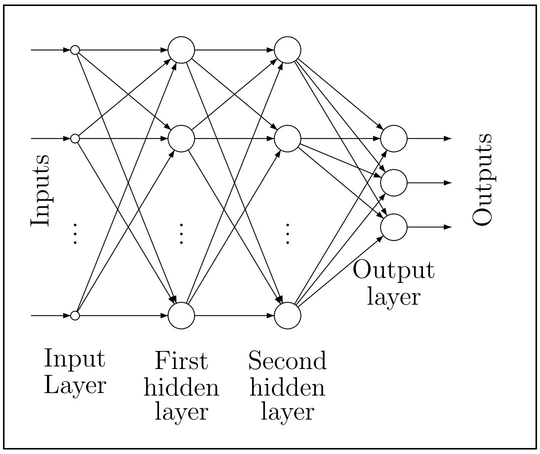

Fig. 1. Multi-layer neural network

Training the neural network means readjusting weights and bias to minimize the sum of released errors:

$$E(f) = \sum_{i=1}^{n} |f(x_i) - c_i|^2.$$

The tuning process of these parameters is described in details in [11, 8].

2.2 Semantic Network

In the 70s Winston [30] proposed a knowledge model intended to allow supervised learning of shapes. It was then adopted in many knowledge based systems and took the name of semantic network.

A semantic network is constituted of two main elements: entities represented graphically by nodes and relations symbolised by edges. Generally, a label is associated with each edge to identify the kind of the relationship. Figure 2 is a graphical representation of a semantic network which represents the meaning of the sentence X *shows* Y.

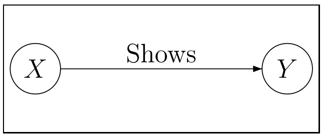

Fig. 2. Semantic network of the predicate Shows(X, Y)

When even the example above typifies a binary predicate, semantic networks are easily used to represent unary predicates. Let us consider the unary predicate *man(Jim)*. Figure 3 shows the corresponding semantic network.

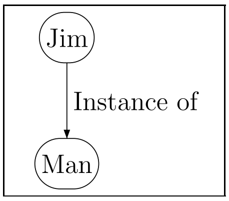

Fig. 3. Semantic net of the predicate Man(Jim)

n-ary predicates are as well representable [15]. Figure 4 shows an example of a semantic network of the predicate *Gives(John, Beggar, $10)*.

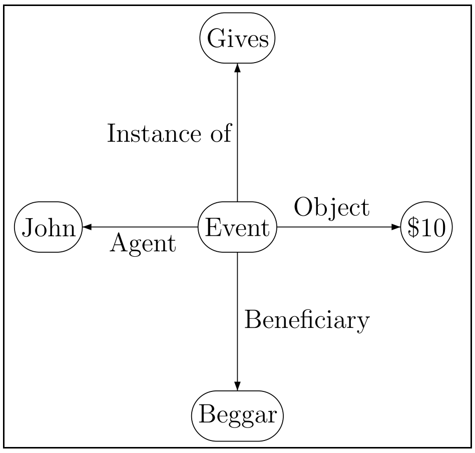

Fig. 4. Semantic net of the predicate *Gives(John, Beggar, $10)*

It is effortless to notice that the n-ary predicates are transformed into binary ones before being represented. The predicates of the last example were transformed into the following set of predicates:

- *Agent(John, Event)*
- *Beneficiary(Beggar, Event)*
- *Object($10, Event)*
- *Is a(Gives, Event)*

Consequently, use of two predicates is not unique.

When the relations represent a partial order and the transitivity is satisfied, the network representing this kind of relations is known as inheritance system. Having this property, the arcs drawing up the relations deduced by transitivity are omitted. Figure 5 illustrates this kind of networks.

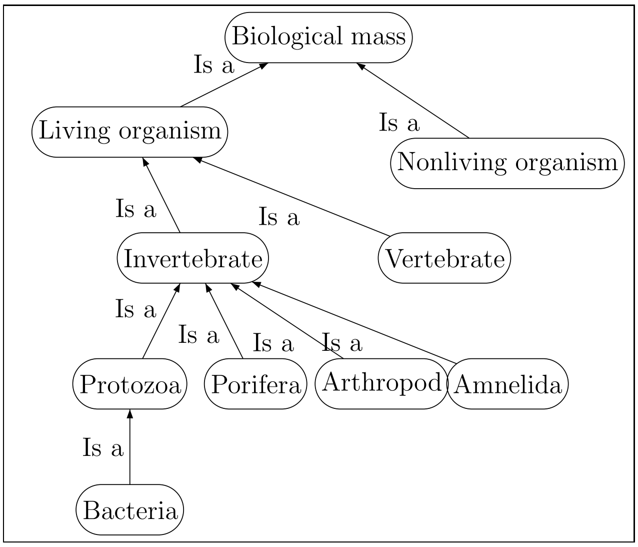

Fig. 5. An inheritance system

Let us introduce the following assertions:

1. We will note the unconditional absolute links using a continuous arrow and the default links using discontinuous arrows $\dashrightarrow$.
2. nodes symbolise predicates or constants
3. no edge must point towards a constant
4. the absolute well as default links can point from a constant
5. when p is a constant, an edge $p \to q$ denotes the predicate $q(p)$

6. else:
$$\forall X, (p(X) \Rightarrow q(X))$$
when p and q are predicates
7. the edge $p \nrightarrow q$ denotes $\neg q(p)$ when p is a constant, and
$$\forall X, (p(X) \nRightarrow q(X))$$
when p and q are predicates
8. the following assertions are equivalent
$$\forall X(p(X) \nrightarrow q(X))$$
$$\forall X(p(X) \rightarrow \neg q(X))$$
$$\forall X(\neg p(X) \rightarrow \neg q(X)).$$

Consequently, inference rules with the absolute links are as follows:

1. **symmetry:** if $p \nrightarrow q$, where p is not a constant, then $q \nrightarrow p$.

   Proof. By definition we have:
$$p \nrightarrow q \Rightarrow p \rightarrow \neg q,$$
   and by definition of the implication:
$$\neg p \vee \neg q$$
   which can be written using the symmetry of $\vee$:
$$\neg q \vee \neg p$$
   using the definition of the implication:
$$q \rightarrow \neg p$$
   then :
$$q \nrightarrow p$$
□

2. **positive field:** If $p \rightarrow q \rightarrow r \rightarrow s$ then $p \rightarrow s$. The proof is rather obvious using transitivity.
3. **negative links:** If $p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_k$ and $q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_m$ and $p_k \nrightarrow q_m$, then $p_1 \nrightarrow q_1$, where q_1 is not a constant.

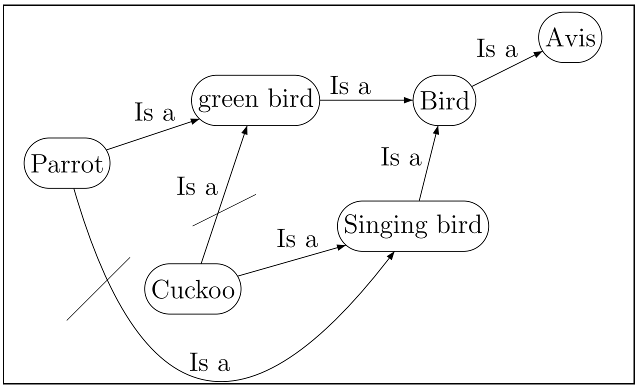

Fig. 6. A semantic network with negative links

Example 1. Let us consider the semantic net of Figure 6.

The canonical interpretation of the network is as follows:

- Green-bird(Parrot)
- Singer-bird(Cuckoo)
- Bird(Green-bird)
- Bird(Singer-bird)
- Avis(Bird)
- ¬Singer-bird(parrot)
- ¬Green-bird(Cuckoo)

From the above description we can use the inference rules to deduce the following links:

1. Singer-bird$\nrightarrow$ parrot
2. Parrot$\rightarrow$ avis

Suppose that the link Green-bird$\nrightarrow$Singer-bird. Using the last rule of inference we obtain parrot$\nrightarrow$Cuckoo.

Indeed, we use $\dashrightarrow$ in order to avoid contradictions.

Example 2. Let us consider the semantic network of Figure 7.

Using the inference rules of the absolute links; by transitivity, we can deduce that Tweety(X)$\rightarrow$ Fly(X) as well as Tweety(X)$\nrightarrow$ Fly(X).

Figure 8 shows how this contradiction is avoided using default links.

Has-wings (X)$\dashrightarrow$ Fly(X) means that flying is the default property of any bird and that Diseased(X)$\nrightarrow$ Fly(X) is a special case which may affect the default property.

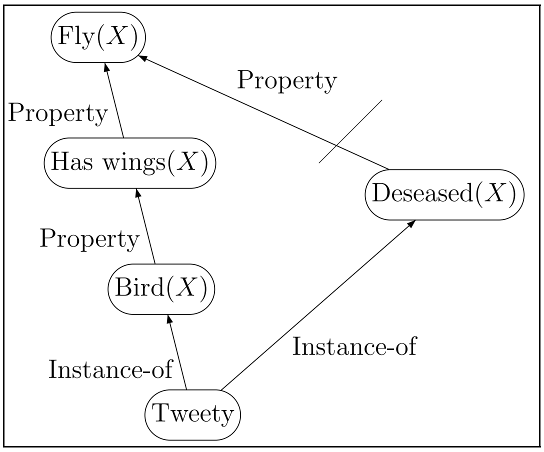

Fig. 7. Semantic network with contradiction

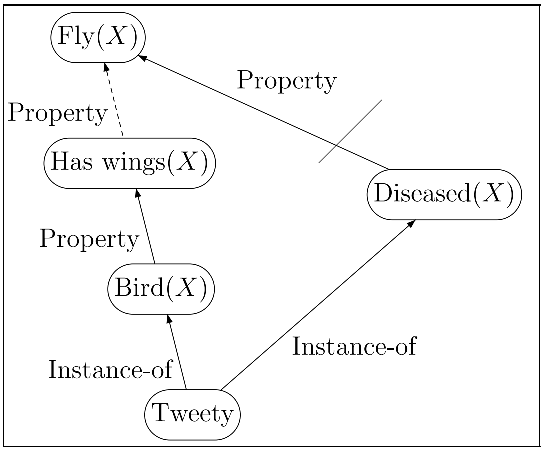

Fig. 8. Default and absolute links

3 DISCUSSION

We have seen two famous models in the previous section. The first belongs to the class of machine learning models while the second belongs to the most known ones in knowledge representation. Table 1 draws up some differences between these models; each of these differences may be an advantage (denoted by ✓) or a drawback.

Many of these drawbacks are related, and may be generalized to many other knowledge models and machine learning structures. Our aim is to define a simple model which constitutes a compromise between the knowledge models and machine

	Neural nets		**Semantic nets**
	Very low level of abstraction	✓	High level of abstraction
	Is not easily interpretable	✓	Is obvious to interprete
✓	Allow unsupervised learning		Should be built by an expert
	Errors are not directly pointed out	✓	Error may be directly pointed out
	Is not extendable	✓	May be extended
	Limited learning capability	✓	Its structure allows addition of new knowledge
✓	Reflect the uncertainty concept		No representation of uncertainty

Table 1. Some differences between neural networks and semantic networks

learning structures and which inherits from the first and the second their advantage, namely, the high level abstraction and the uncertainty representation.

4 THE PRIMARY VERSION

Based on some experiments on human cognition behaviours, a simple model has been proposed in [12] where knowledge is represented by entities linked using binary relations. Each relation (or entity) is weighted with the number of simultaneous occurrences of the the relation components (or entity). Weights are said remembering level.

The knowledge base may be viewed as a non oriented graph $G(N, A)$ where N is a set of nodes, $N = \{(x, y) \mid x \text{ is a label and } y \text{ is the remembering level}\}$. A is the set of weighted edges $A = \{(x_1, x_2, y) \mid x_1, x_2 \text{ are labels of entities, } y \text{ is the weight of the relation } (x_1, x_2)\}$. Figure 9 shows an illustration of the graphical representation of the sentence "*parasympathomimetic is a word*". w_is are weights that correspond either to an entity or relation. In case of relations, the weight corresponds to the number of simultaneous occurrences of the adjacent entities. Weight of an entity refers to the number of its occurrences either with other entities or alone.

4.1 Implementation

The first implementation of the model above was in Arabic script recognizer [24]. Our application was based on the extraction of characteristics of hand-written Arabic letters. The encoding method is summarized in the Algorithm 4.1.

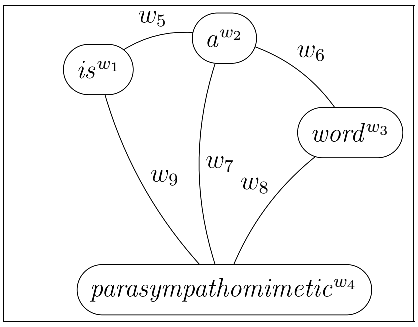

Fig. 9. Graphic representation of the sentence "*parasympathomimetic is a word*"

In each sub-area, each axis will carry out 3 letters. The code extracted from the letter (Figure 11) is as follows: 0-m-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-c, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-c-0, 0-m-0.

Algorithm 1 Encoding of the letter graph

```
1: Divid horizontally the writting area into three parts : $Middle$, $Pole$ and $jamb$
   (Figure 10).
2: Cross the letter graph by vertical axis outstripped by a fixed distance as shown
   in (Figure 11)
3: for all axis do
4:    for all parts of the axis (i.e. $Middle$, $Pole$ and $jamb$) do
5:       Let $n$ be the number of intersection points.
6:       if $\geq n \leq 9$ then
7:          concatenate the code with the letter '$c$'.
8:       else
9:          if $10 \geq n \leq 40$ then
10:            concatenate the code with the letter '$m$'.
11:         else
12:            if $n > 40$ then
13:               concatenate the code with the letter '$l$'.
14:            end if
15:         end if
16:      end if
17:   end for
18: end for
```

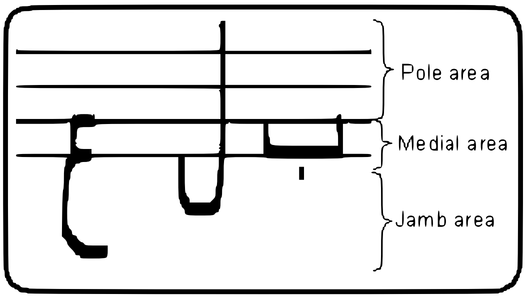

Fig. 10. The three writing areas

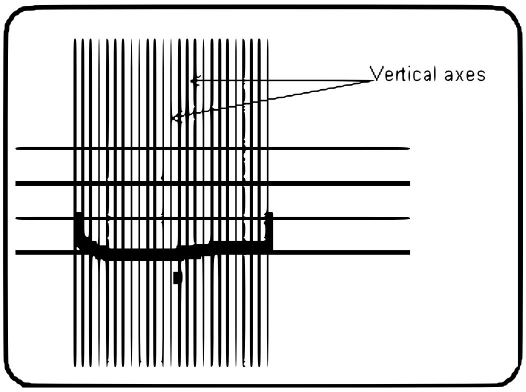

Fig. 11. Code generation

4.1.1 Learning Phase

The set of generated codes for all graphs may be viewed as a language for which we can build a classical finite automaton as the first step recogniser. Hence, we implemented a simple deterministic finite automaton to check the common shapes (Figure 12).

In practice every person has his/her own style to write the same character and the code generated from the graph will differ from that recognized by the automaton. In such case our system learns the scripting manner of the author of the non-recognized letter. When the automaton fails to recognize the graph, the recognizer prompts the user to give formally the letter which corresponds to the graph (for example ب). The code generated from the unrecognised graph will form the entities linked to the letter entity. Since the model is initially empty when the code occurs for the first time, this code is added to the model with the weight 1. The same thing is done with the entity which represents the letter ب. The conjunction be-

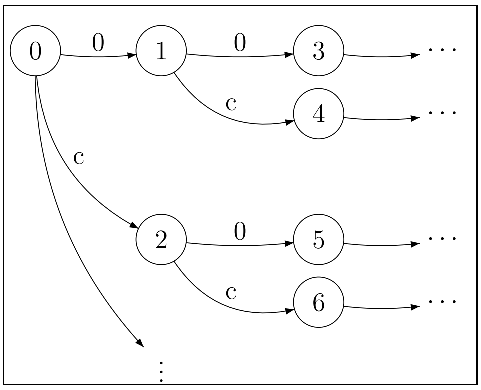

Fig. 12. A portion of the automaton

tween these two entities, graph code and the drawn letter, is assigned with weight 1 if the conjunction is added for the first time, otherwise the conjunction weight is incremented. In this phase, the system learns the writer's scripting manner. Once the model is sufficiently enriched, the learning phase is stopped.

4.1.2 Recognition Using the Model

The system tempts to recognise the drawn letter using an automaton; if the automaton fails to recognise the graph, the recognizer looks for the code in the knowledge base that is the closest to the generated code by the graph. The following Table shows some test results.

Letters	Number of correct recognitions/5 tests	Number of codes
ب	5	1
ك	4	29
د	5	2
ت	5	1
ث	4	11
و	3	11
ق	4	7
ف	4	4

5 THE NEW VERSION

Let $X = \langle x_1, \ldots, x_n \rangle$ be a vector of n features. A classification system assigns X to some class C_i according to the values of the x_ps. It can be viewed as a function f which maps vectors in $D_1 \times D_2, \ldots \times D_n$ to some class in the set of all possible classes $\mathcal{C}$. In machine learning, we try to build semi- or quasi-automatically an approximation of the function f [16, 23, 13]. For simplicity, let us consider a Boolean classification, where the system has to choose between just two classes 0 or 1. Values of the features x_ps also belong to $\{0, 1\}$. Let f be a Boolean function where $f(x_1, x_2) = x_1 x_2$. f assigns to $\langle 0, 1\rangle$, $\langle 1, 0\rangle$, $\langle 0, 0\rangle$ the class 0, and only the vector $\langle 1, 1\rangle$ is assigned to the class 1. We will say that *in the context f, there are two entities 0 and 1*, and *in the context 1, the only available relationship is* $\langle 1, 1\rangle$. All other relationships belong to the context 0. Figure 14 gives a simple set representation of f. Note that we can represent "*context of*" by a relation between the context and the relation belonging to. Since the entity 0 (or 1) has the same semantic being either an input or an output of the function, we use a unique representation of 0 (or 1). Figure 13 is a graphical representation of the instance of the model which represents f.

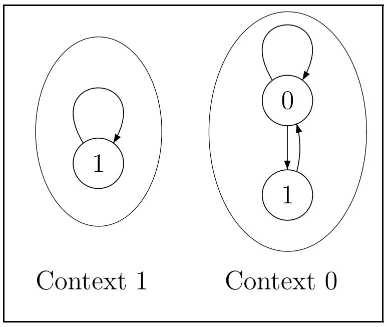

Fig. 13. Context representation of the the function $f = x_1 x_2$

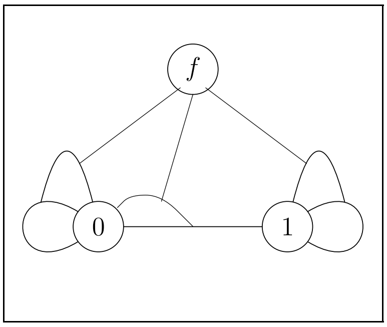

Fig. 14. Model instance of f

In Figure 14, edges are not oriented because the *and* operator is symmetric, but also because the reasoning can be done in two ways. Having the output value (context), we can find the input values that lead to; we also can find deterministically the value of the output when we have the input values.

Let us examine how to add a new knowledge to the instance of Figure 14. Let f' be a Boolean function where $f'(x_1, x_2, x_3) = f(x_1, x_2) + x_3$. A canonical representation of the new instance of the model is given in Table 2, and the graphical representation is shown in Figure 15.

R_0 :	1	$\leftrightarrow$	1
R_1 :	$R0$	$\leftrightarrow$	1
R_2 :	0	$\leftrightarrow$	0
R_3 :	0	$\leftrightarrow$	1
R_4 :	R_2	$\leftrightarrow$	0
R_5 :	R_3	$\leftrightarrow$	0
R_6 :	R_4	$\leftrightarrow$	f
R_7 :	R_5	$\leftrightarrow$	f
R_8 :	f	$\leftrightarrow$	1
R_9 :	1	$\leftrightarrow$	R_8
R_{10}:	f	$\leftrightarrow$	0
R_{11}:	f	$\leftrightarrow$	R_{10}
R_{12}:	f'	$\leftrightarrow$	R_9
R_{13}:	f'	$\leftrightarrow$	R_{11}

Table 2. Canonical description of f and f'

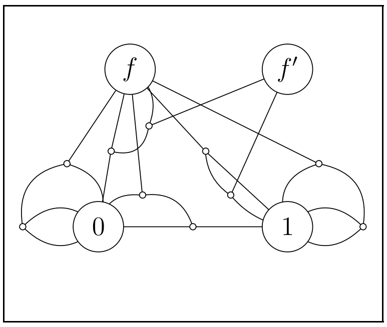

Fig. 15. Model instance of f and f'

As opposite to semantic networks which explicitly label relations with their semantics, the philosophy of our model is that "*the semantic of things is expressed in an architecture of links*". Note also that we may have different semantics for the same relation in different contexts. Entities may be contexts as well. Actually, in our model, a context may be an entity of a context and relation may be a component of another one. For instance, the relation R_{13} means that in the context f', the output is f when the third parameter is equal to 0. Note that we represented the new function f' which really depends on f, using knowledge on f already represented.

6 INFERENCE RULES

When the representation contains so much knowledge on some context, inference is then a link-follow-like task. In practice, exhaustive representation is generally impossible either because the complete description of some part of the real world isn't available, or because the computational resources might be burdened by maintaining such a big volume of information.

Generalization will refer to the use of the available entities and relations in an instance of the model to generate new relations and/or entities.

Finite automata model remains a fascinating one even in artificial intelligence. The following example shows how an instance of our model may be built up and trained on a formal knowledge such finite automata. First, let us recall that a deterministic finite automaton is a simplified model of neural networks especially adapted to formal languages recognition. Briefly, DFA defines a set of transitions due to the occurrence of a pair $(r, a) \in Q \times \Sigma$, where Q is a finite set called set of states, and Σ set of symbols; towards an other state s. Each state may be final or not.

Formally spoken.

Definition 1. Let A be a finite set of symbols. A finite automaton is a quintuple $\mathcal{A} = (Q, A, \delta, q_0, F)$ where

- Q is a finite set of states,
- A is the alphabet,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states, and
- $\delta \subseteq Q \times (A \cup \{\varepsilon\}) \times Q$ is the transition mapping;

we shall denote, for $p \in Q$, $a \in A \cup \{\varepsilon\}$, $\delta(p, a) = \{q \in Q \mid (p, a, q) \in \delta\}$. The size of an automaton $\mathcal{A}$, denoted by $|\mathcal{A}|$, is the number of its states. The automaton $\mathcal{A}$ is called *deterministic* if $|\delta(q, a)| \leq 1$, $\forall q \in Q$, $\forall a \in A$.

Definition 2. A *path* in $\mathcal{A}$ is a sequence (q_i, a_i, q_{i+1}), $i = 0, \cdots, n$, of consecutive transitions. Its *label* is the word $w = a_1 a_2 \cdots a_n$.

Definition 3. A word $w = a_1 a_2 \cdots a_n$ is *recognized* by the automaton $\mathcal{A}$ if there is a path labelled w such that $q_{n+1} \in F$.

Definition 4. The language *recognized* by the automaton $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$, is the set of words it recognizes.

Definition 5. The right language of a state q in the automaton $\mathcal{A}$, denoted by $\vec{\mathcal{L}_q}(\mathcal{A})$, is obtained by setting q to be the initial state $\mathcal{A}$, i.e., $\vec{\mathcal{L}_q}(\mathcal{A}) = \{w \in A^* \mid \delta(q, w) \cap F \neq \emptyset\}$.

Definition 6. Two states p and q in Q are said to be equivalent if and only if $\vec{\mathcal{L}_p}(\mathcal{A}) = \vec{\mathcal{L}_q}(\mathcal{A})$

Details on finite automaton can be found in many books on theory of formal languages [10]. Minimization of a finite automaton $\mathcal{A}$ is the building of an eventually different deterministic finite automaton $\mathcal{B}$ in which no pair of states are equivalent. The problem is then to find every equivalent pair of states in $\mathcal{A}$, or in a negative reasoning, to find all non equivalent pairs. The minimization process starts with the assumption that none of the final states is equivalent to a not final one.

$$\forall p \in F \text{ and } q \in Q - F : \vec{\mathcal{L}_p}(\mathcal{A}) \neq \vec{\mathcal{L}_q}(\mathcal{A})$$

In fact, all minimization algorithms are based on the principle of finding the partition of Q using this last hypothesis. Albeit in [27, 28, 29] B. Watson assumes that Brzozowski's algorithm [1] isn't based on the partition of the set Q. In [3] we prove that even this algorithm is based on the same partitioning principle.

Let us consider the deterministic finite automaton of Figure 16.

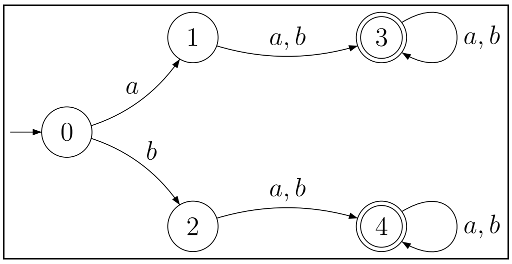

Fig. 16. Deterministic finite automaton to be minimized

Performed by a machine or manually, an algorithm uses:

1. basic knowledge on finite automaton graphs such as:
   (a) an edge label is a symbol
   (b) a circle is a state
   (c) a double circle is a final state others are not final
   (d) an edge is a transition
2. knowledge on state's equivalence:
   (a) a final state is not equivalent to a non-final one
   (b) if at least two transitions map with the same symbol two states s_1 and s_2 into two non equivalent ones, then s_1 and s_2 are not equivalent.

Figure 17 is the representation of the basic knowledge on finite automata.

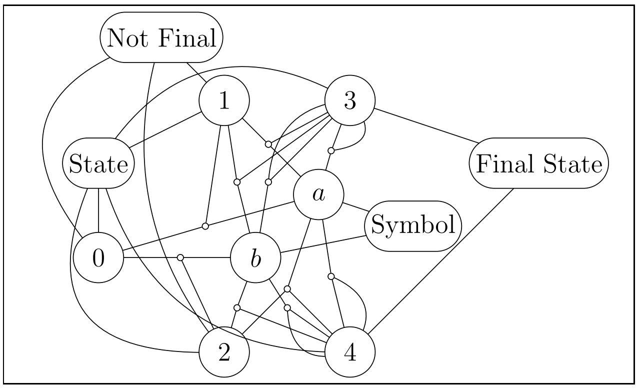

Fig. 17. Representation of the automaton of Figure 16

At the first view, the representation seems to be too heavy. Access to some knowledge is an access to only a part of the global model. For instance, entities a and 1 are bounded up with a relation which is also bounded with the entity 3. This means that when the pair $(1, a)$ occurs, the automaton will transit to state three. The instance of the model in Figure 17 shows the representation of the basic knowledge on the DFA (Figure 12). In classical programming, such structure is an input of the minimization algorithm. We have now to define the reasoning rules to generate knowledge on states equivalence. Figure 18 represents the rule that "*a final state is not equivalent to a non-final one*". Using this rule, if a part of the instance contains at least links in the left side of Figure 18, whatever are values of x_1 and x_2, we can substitute it by the right side representation.

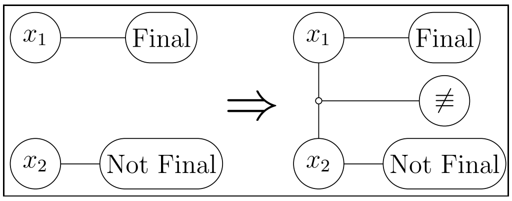

Fig. 18. First rule of knowledge generation in the instance of Figure 17

Figures 19 and 20 are the representation of the rule "*if two states go with the same symbol to a non-equivalent pair of states then the two first states are not equivalent*". This has two forms. The first one is represented in Figure 18 where x_1, x_2, x_3, x_4 are states and x_5 is a symbol. x_3 and x_4 are already known to be non-equivalent which results in that x_1 and x_2 are non-equivalent too. The second form is showed in Figure 19: x_1, x_3, x_4 are states and x_5 is a symbol. x_3 and x_4 are already known to be non-equivalent. Since x_1 goes to x_3 with the symbol x_5 and x_3

goes to x_4 with the same symbol, we can deduce that x_1 and x_3 are non-equivalent too (the right part of the figure).

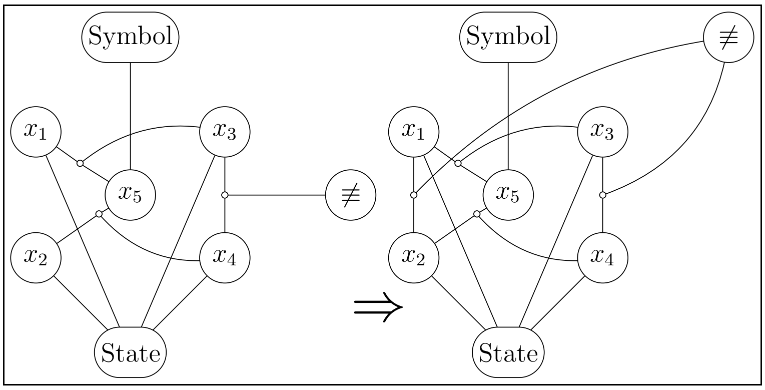

Fig. 19. First form of the first knowledge generation rule

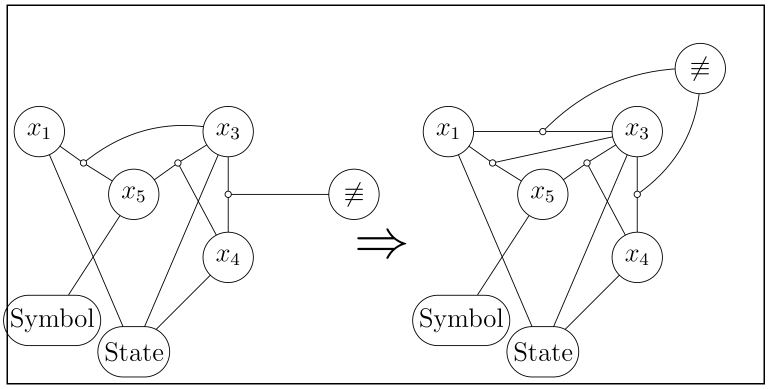

Fig. 20. Second form of the second knowledge generation rule

After the application of the first rule (Figure 18) on the pair of states 1 and 3, the model becomes as shown in Figure 21.

Before the generation of the entity $\not\equiv$ using the first rule, the model instance didn't contain a sufficient knowledge to use the second rule. Figure 22 shows an example of the generation of a new knowledge using the second rule (Figure 20).

7 UNCERTAINTY AND UNSUPERVISED LEARNING

In the previous section, we saw how to define a basic knowledge and inference rules in an explaining-like manner. The purpose of this section is to show the unsupervised

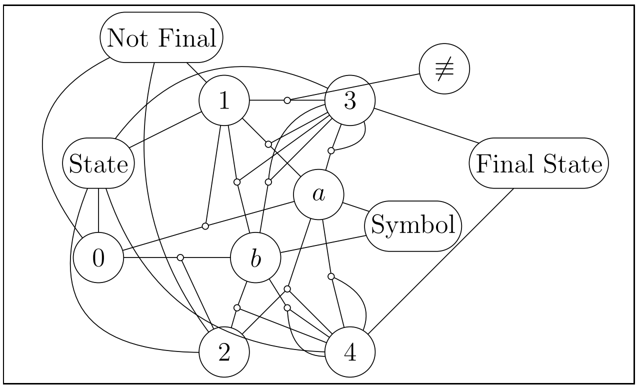

Fig. 21. New knowledge generated in the instance of Figure 17 after the application of the first rule

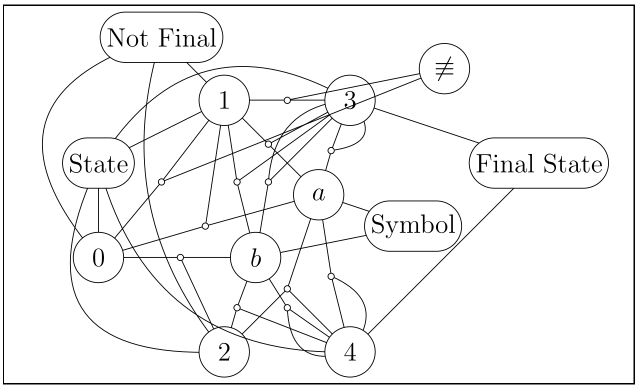

Fig. 22. Example of the application of the second rule

learning using the same model. Generally, a system is called an unsupervised learning system when it can acquire automatically the knowledge issued from information contained in some sample. Such observation generates an uncertain knowledge. To see clearer, let us consider a sample of 100 birds. 24 are ill, and two have their wings broken and can not fly. 10 of the 24 ill birds can fly while the others can not. All other birds can fly. In this example we consider two classes: '*can fly*' and '*can not fly*'. Figure 23 is a representation of the knowledge issued from the observation of this sample.

Notice that both entities and relationships are weighted with occurrence numbers. For instance, the occurrence number of the relationship between the entity *Wings* and *Broken* is weighted with the number of simultaneous occurrences of both.

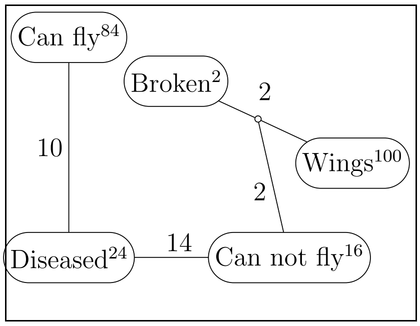

Fig. 23. Knowledge representation issued from the observation of a sample of 100 birds

The relationships of the entity *Diseased* may carry out a contradiction without weights, since it is linked to the context *Can fly* and *Can not fly.* Let us now see how a classification of a new bird should be performed using the knowledge represented in Figure 23. Suppose that the bird to be classified is ill and has safe wings. We can then say that the probability of being in the class *Can fly* is $\frac{10}{24} \approx 0.42$.

Let $\mathcal{E}$ be a set of l samples represented in the form of vector $X = \langle x_1, \ldots, x_n \rangle$, where x_is are values of some attributes. Suppose we know that $C_0 nb$ vectors belong to class C_0. Let $Y = \langle y_1, \ldots, y_k \rangle$ be a new vector where $Y \notin \mathcal{E}$. Note that the size of Y is not n as Xs. We will consider that Y does not contain all and just the n attributes. Classification should thus be performed with relatively incomplete information to our observations. The advantage to have such modular representation where modules are mutually independent is in being able to do tasks such as classification with incomplete information. Formally spoken, we have:

$$P(C_0|Y) = \frac{P(C_0 \cap Y)}{P(Y)}.$$

$P(C_0|Y)$ is the probability that the class C_0 occurs when the attributes compounding the vector Y occur. We have:

$$P(C_0|Y) = \frac{P(C_0 \cap y_1 \cap \ldots \cap y_k)}{P(y_1 \cap \ldots \cap y_k)}$$

which we can reformulate as follows:

$$P(C_0|Y) = \frac{P((C_0 \cap y_1) \cap (C_0 \cap y_2) \cap \ldots \cap (C_0 \cap y_k))}{P(y_1 \cap \ldots \cap y_k)}.$$

Even if it is wrong we will suppose as a Bayesian classifier [21, 25] does that occurrence of any attributes is independent from all others. Upon this hypothesis we can write:

$$P(C_0|Y) = \frac{P(C_0 \cap y_1)P(C_0 \cap y_2)\dots P(C_0 \cap y_k)}{P(y_1)\dots P(y_k)}.$$

In our model, $P(C_0 \cap y_i)$ should be approximated with the weight of the relationship between the entities C_0 and y_i. Note that we can use the approximation in the form $P(C_0 \cap y_i \cap y_j)$ when the model instance contains the relationships between the entity C_0 and the relationship $\langle y_i, y_j\rangle$ which uses the following formula:

$$P(C_0|Y) = \frac{P(C_0 \cap y_1 \cap y_{i_1})P(C_0 \cap y_1 \cap y_{i_2})\dots P(C_0 \cap y_k \cap y_{j_l})}{P(y_1 \cap y_{i_1})\dots P(y_k \cap y_{j_l})}$$

where $P(u \cap v)$ is approximated by weight $(\langle u, v\rangle)$, which is the weight of the relationship between u and v; recall that u and v may be either entities or relationships. For example, if the model instance contains $R_1 = \langle x_1, y_1\rangle, R_2 = \langle x_2, y_2\rangle, R_3 = \langle R_1, R_2\rangle$, then $P(x_1 \cap x_2 \cap x_3 \cap x_4)$ is supposed to be the weight of R_3.

The approximation can use an intersection of any size when relationships linking their entities are represented.

8 INFERENCE

In this section, we show how an inference can be performed using our structure. Let us denote the representation of some feature x $M(x)$. R_i will denote any relation in the model instance and Context(R_i) will denote the entity which is the direct context of R_i. Algorithm 2 summarizes the inference steps on an instance built using deterministic knowledge.

Algorithm 2 Inference on deterministic knowledge

```
Require: $e = \{e_1, \dots, e_l\}$ Inputs should be represented in the instance of the model
Ensure: $e' = \{e'_1, \dots, e'_m\}$ Outputs are some other entities bounded up with inputs
  for all $e_i$ do
    activate $M(e_i)$
  end for
  for all $R_j = \langle M(e_i), M(e_k)\rangle$ where $e_i, e_k \in e$ do
    activate $R_j$ //activate all relations compounded from inputs
  end for
  repeat
    activate Context($R_j$)
    for all $R_u = \langle c_v, c_w\rangle$ where $c_v, c_w$ are contexts recently activated do
      activate $R_u$ //activate all relations between contexts recently activated
    end for
  until Stability
```

More intuitively, in the case of deterministic knowledge, inputs are entities represented in an instance of the model. The algorithm proceeds as follows:

1. All inputs' representations and direct relations between them are activated.
2. All contexts of direct relations are activated.
3. Repeat 1 and 2 considering contexts recently activated as inputs until there is no new activated context.

   Let us consider the following assertions:

- each entity x (characteristic or context) is represented in the model instance by $M(x)$ which is its graphical representation and $Weight(M(x))$ the weight of its representation;
- each relation R_i in the model has three components: two representations of its entities and the weight of the relationship. Say $R_i = \langle M(x_1), M(x_2), \omega\rangle$;
- the context of any relation is denoted by $Context(R_i)$;
- before being activated, entity (or relation) has a null activation weight $AWeight(M(x_i)) = 0$ (or $AWeight(R_i) = 0$), and once activated $AWeight(M(e_i))$ (or $AWeight(R_i) = 0$) is assigned with a non null value.

Algorithm 3 describes the inference in a model of an uncertain knowledge:

Algorithm 3 Inference on uncertain knowledge

```
Require: $e = \langle e_1, \dots, e_p\rangle$
Ensure: $e' = \langle e'_1, \dots, e'_m\rangle$
  for all $e_i$ do
    activate $M(e_i)$ with weight $Wheight(M(e_i))$ say $AWheight(M(e_i)) \leftarrow Wheight(M(e_i))$
  end for
  repeat
    for all $R_j = \langle M(e_i), M(e_j), \omega\rangle$ such that $AWheight(M(e_i)) > 0$ and $AWheight(M(e_j)) > 0$ do
      $AWheight(R_j) \leftarrow \min(AWheight(M(e_i)), AWheight(M(e_j)), Wheight(R_j))$
      //activate the relation with minimum value between $AWheight(M(e_i)), AWheight(M(e_j)), Wheight(R_j)$
    end for
  until nothing
```

9 CONCLUSION

Performance of any knowledge based system is, mainly due to its knowledge base, in terms of containment and representation. That knowledge has two known sources: either a human expert or an observation. Our aim was to design a flexible model which can support both modes of learning. The first mode requires a simple comprehensible representation to allow an easy expression. The second requires an efficient representation of knowledge uncertainty. A purification process that minimizes the representation may be done in the case of unsupervised learning. This is possible by eliminating the insignificant relationships and entities. When relationships going from an entity to two opposite entities are weighted with approximately the same weight, this intuitively means that the first entity isn't a good discrimination. At the first view our model seems to be Bayesian nets-like one. Let us recall that the last one implements directly the conditional probabilities on events. When a new event occurs, an update requires another computation of all probabilities of all bound events. Our model adds the new events intuitively by incrementing weights of individual entities and relations binding them. In other terms, the Bayesian nets model makes computation on knowledge, then represents it; our model represents and lets computation to the time due. The main idea of our model is to keep a basic information (occurrence number) to allow any kind of computation on.

REFERENCES

[1] BRZOZOWSKI, J.: Canonical Regular Expressions and Minimal State Graphs for Definite Events. In Mathematical Theory of Automata, pp. 529–561. Polytechnic Press, Polytechnic Institute of Brooklyn, N. Y., 1962. Vol. 12 of MRI Symposia Series.

[2] BUNTINE, W.: Learning Classification Trees. In D. J. Hand, editor, Artificial Intelligence Frontiers in Statistics, pp. 182–201. Chapman & Hall, London, 1993.

[3] CHAMPARNAUD, J.-M.—KHORSI, A.—PARANTHOËN, T.: Split and Join for Minimizing: Brzozowski's Algorithm. In M. Balík, M. Šimánek, editors, Proc. of Stringology Conference, pp. 96–104, CTU, 2002. Research Report DC-2002–03.

[4] DAVIS, R.—SHROBE, H.—SZOLOVITS, P.: What Is a Knowledge Representation? AI Magazine, Vol. 14, 1993, No. 1, pp. 17–33.

[5] DE MÁNTARAS, R. L.: A Distance-Based Attribute Selection Measure for Decision Tree Induction. Machine Learning, Vol. 6, 1991, pp. 81–92.

[6] GAROFALAKIS, M. N.—HYUN, D.—RASTOGI, R.—SHIM, K.: Efficient Algorithms for Constructing Decision Trees with Constraints. In Knowledge Discovery and Data Mining, 2000, pp. 335–339.

[7] GLICK, M.—RUMELHART, D.: Neuroscience and Connectionist Theory. The Development in Connectionist Theory, Erlbaum Associates, Hilldsdale, NJ, 1989.

[8] HAYKIN, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall, 1998.

[9] HOLLAND, J.: Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press, 1975.

[10] HOPCROFT, J. E.—ROTWANI, R.—ULLMAN, J. D.: Introduction to Automata Theory, Languages and Computability. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[11] KECMAN, V.: Learning and Soft Computing. The MIT Press, 2001.

[12] KHORSI, A.: A New Model for Knowledge Representation and Automatic Deduction for AI Applications. In Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics, ISAS-SCIs 2001, July 2002.

[13] Machine Learning Journal, Vol. 5, 1993, No. 6. Special issue on Learning and Discovery in Databases.

[14] McCulloch, W. S.—Pitts, W. H.: A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, Vol. 5, 1943, pp. 115–133.

[15] Nilsson, N. J.: Principles of Artificial Intelligence. Springer-Verlag, Berlin, 1990.

[16] Nilsson, N. J.: Introduction to Machine Learning. An Early Draft of a Proposed Text Book, 1996.

[17] Quinlan, J. R.: Induction of Decision Trees. In J. W. Shavlik and T. G. Dietterich, editors, Readings in Machine Learning, pp. 57–69. Kaufmann, San Mateo, CA, 1990.

[18] Rastogi, R.—Shim, K.: Public: A Decision Tree Classifier That Integrates Building and Pruning. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, VLDB '98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24–27, 1998, New York City, New York, USA, pp. 404–415. Morgan Kaufmann, 1998.

[19] Rosenblatt, F.: Principles of Neurodynamics. Spartan Books, Washington, 1958.

[20] Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. In J. W. Shavlik and T. G. Dietterich, editors, Readings in Machine Learning, pp. 138–149. Kaufmann, San Mateo, CA, 1990.

[21] Sahami, M.: Learning Limited Dependence Bayesian Classifiers. In Second International Conference on Knowledge Discovery in Databases, 1996.

[22] Shavlik, J.—Mooney, R.—Towell, G.: Symbolic and Neural Learning Algorithms: An Experimental Comparison. Machine Learning, Vol. 6, 1991, pp. 111–143.

[23] Shavlik, J. W.—Dietterich, T. G.: General Aspects of Machine Learning. In J. W. Shavlik and T. G. Dietterich, editors, Readings in Machine Learning, pp. 1–10. Kaufmann, San Mateo, CA, 1990.

[24] Meslati, L. S.—Sellami, M.: Perceptual Recognition of Arabic Literal Amounts. Computing and Informatics, Vol. 25, 2006, No. 1, pp. 43–59.

[25] Vailaya, A.—Figueiredo, M.—Jain, A. K.—Zhang, H. J.: Content-Based Hierarchical Classification of Vacation Images. Technical Report MSU-CPS-99-9, Department of Computer Science, Michigan State University, East Lansing, Michigan, February 1999.

[26] Valls, J. M.—Galván, I. M.—Isasi, P.: Improving the Generalization Ability of RBNN Using a Selective Strategy Based on the Gaussian Kernel Function. Computing and Informatics, Vol. 25, 2006, No. 1, pp. 1–15.

[27] Watson, B. W.: A Taxonomy of Finite Automata Construction Algorithms. Computing Science Note 93/43, Eindhoven University of Technology, The Netherlands, 1993.

[28] Watson, B. W.: A Taxonomy of Finite Automata Minimization Algorithms. Computing Science Note 93/44, Eindhoven University of Technology, The Netherlands, 1993.

[29] Watson, B. W.: Taxonomies and Toolkits of Regular Language Algorithms. Ph. D. thesis, Eindhoven University of Technology, the Netherlands, 1995.

[30] Winston, P. H.: Learning Structural Descriptions From Examples. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1970.

Ahmed Khorsi is a research member in the Laboratory of Mathematics of Djillali Liabes University, and has been a computer science teacher at several universities since 2001. He provided computer science courses and labs at Djillali Liabes University, Moulay Tahar University Center, and others. He is currently finalizing his Ph. D. study. He obtained his Bachelor and Master degrees in 2000, 2002, respectively from Djillali Liabes University. He communicated papers on finite automata theory and artificial intelligence at several international conferences. He has acquired a valuable experience in many practical areas of computer science, especially in Unix, Linux and open source programming, and participated as a teacher in Linux summer schools.