Probabilistic Approach for Reduction of Irrelevant Tree-structured Data

Amaury Habrard, Marc Bernard, Marc Sebban

EURISE – Université Jean Monnet de Saint-Etienne
23, rue du Dr Paul Michelon – 42023 Saint-Etienne cedex 2 – France
{amaury.habrard,marc.bernard,marc.sebban}@univ-st-etienne.fr

Abstract. This article aims at pruning noisy or irrelevant subtrees in a set of trees. The originality of this approach, in comparison with classic techniques in prototype selection, comes not from the non-deletion of the whole tree, but rather of some of its subtrees. Our method is based on the computation of confidence intervals on a set of subtrees according to a probability distribution. We propose an approach to assess these intervals on this specific type of data and show experimentally its interest in the context of learning from noisy data.
Keywords. data reduction, prototype selection, tree-structured data, noisy data

1 Introduction

The use of structured or semi-structured data is increasing in research domains such as knowledge discovery in databases or machine learning. For example, a learning sample can be represented by a relational database. The increasing interest for directly exploiting such data has lead to a new field of research named multi-relational learning [1]. XML data are another example of structured representation. They are easily available in huge quantity on the web, and have stimulated the interest for tree-structured data.

The tree representation gives an interesting compromise between graphs and linear representations. Actually, they allow the expression of hierarchical dependences and are less costly for processing than graphs. A lot of recent works are interested in knowledge discovery from tree-structured data. For example we can cite the extraction of frequent trees [2, 3], or the detection of tree patterns, either in the framework of relational databases or in the one of XML data [4, 5]. Trees are also studied in machine learning, particularly in the inference of tree patterns [6, 7], or in tree automata induction [8, 9].

While trees are less complex data structures than graphs, they are nevertheless more difficult to process than linear representations. Consequently, in domains where the amount of data is huge, and the level of noise very high (in data mining for example), it seems interesting to select only relevant data to work on. This way to proceed allows one to reduce not only the complexity of data storage but also the generalization error of the induced models. These tasks

are the matter of data reduction which can be achieved via two ways: prototype selection (PS) [10] and feature selection (FS) [11]. It is important to note that these methods are usually applied to non-structured data and aim at totally deleting either an example (PS) or a feature (FS). However, in the context of tree-structured data, we assume that only some particular subtrees of a given tree are noisy or irrelevant. Then it is not necessary to completely delete this tree. The suppression of a subtree can be seen as an hybrid approach of data reduction. It looks like prototype selection when a tree is completely deleted, and a local feature selection when only subtrees are removed.

In this paper, we propose a probabilistic approach based on confidence intervals to prune irrelevant subtrees. In fact we study the probability of a subtree to be in a class issued from a partitioning of the whole set of subtrees (Section 2). In Section 3, we describe a partitioning method based on tree patterns. Section 4 deals with application of our method both on artificial and real data.

2 Pruning Subtrees

We consider the problem of inferring an estimation model in the presence of noisy tree-structured data. We aim at pruning noisy and irrelevant subtrees before the learning process. Our approach consists in estimating a probability distribution on the set of all the subtrees of the learning sample. We construct a partition of the learning subtrees, and for each element of this partition, we compute a confidence interval containing $(100 - \alpha)\%$ of the considered learning subtrees. Those with a too small probability are pruned.

2.1 Definitions and Notations

A tree has a root node and a set, eventually empty, of children. A leaf is a node without child. Trees, we are interested in, are constructed over a signature. Each node is labeled by a functional symbol, and all the nodes labeled by the same symbol have exactly the same number of children. Finally the symbols are typed and the children ordered.

Definition 1 *A signature Σ is a 4-tuple $(S, V, arity, \sigma)$. S is a finite set whose elements are called sorts. V is a finite set whose elements are called functional symbols. arity is a mapping from V into $\mathbb{N}$, $arity(f)$ is called the arity of f. σ is a mapping from V into S, $\sigma(s)$ will be called the sort of s. We denote Σ^T the set of trees defined relative to a signature Σ.*

The data reduction method we propose is based on a probability distribution defined on a set of subtrees. Thus we introduce some notations related to the subtrees of a sample of trees.

Definition 2 *Let T be a sample of trees. We denote $Sub(T)$ the set of the subtrees of T and $MSub(T)$ the multi-set of subtrees of T.*

For example, in the sample $E = \{h(f(a,b)), f(h(b),a)\}$, $Sub(E) = \{a, b, f(a,b), h(b), h(f(a,b)), f(h(b),a)\}$ and $MSub(E) = \{a, b, a, b, f(a,b), h(b), h(f(a,b)), f(h(b),a)\}$

Definition 3 *A position is a couple (f,p) (denoted $f.p$) where $f \in V$ and $p \in \mathbb{N}$ such that $1 \leq p \leq arity(f)$. A position allows us to design the subtree corresponding to the child number p of the symbol f (the children are ordered from left to right).*

For example, in the sample $E = \{h(f(a,b)), f(h(b),a)\}$, the subtree a is at position $f.2$ of the tree $f(h(b),a)$ and at position $f.1$ of the tree $f(a,b)$.

2.2 Estimation of a Probability Distribution

In a first step, we estimate a probability distribution relatively to a sample of trees T. This distribution then allows us to compute a probability for each subtree in $Sub(T)$. We use a similar approach to the "N-grams" often used as a model of natural language modeling [12]. This approach assumes that the probability of a given symbol in a string, can be computed using the $n-1$ previous symbols. We use a similar principle, with $n = 2$, computing the probability of a symbol relatively to its parent. Note that a different adaptation of this approach in the context of trees has already been proposed in [9]. For each symbol a of the sample, we assess via $\hat{p_c}(a \mid f.i)$ its probability to be the child number i of any symbol f. Formally:

$$\forall a \in V, \forall f \in V, \forall 1 \leq i \leq arity(f),\ \hat{p_c}(a \mid f.i) = \frac{\text{Number of occurrences of } a \text{ in } f.i}{\text{Number of occurrences of } f}$$

Moreover we estimate for each symbol its probability to be the root of a tree.

$$\forall a \in V,\ \hat{p_r}(a) = \frac{\text{Number of examples with } a \text{ for root}}{\text{Number of examples}}$$

Finally the estimation of the probability of a tree $t = f(t_1, \ldots, t_n)$ is computed as follows:

$$\hat{p_a}(f(t_1, \ldots, t_n)) = \hat{p_r}(f) \times \hat{p_{pos}}(t_1 \mid f.1) \times \cdots \times \hat{p_{pos}}(t_n \mid f.n)$$

where $\hat{p_{pos}}$ is recursively[1] defined by:

$$\hat{p_{pos}}(g(u_1, \ldots, u_n) \mid f.n) = \hat{p_c}(g \mid f.n) \times \hat{p_{pos}}(u_1 \mid g.1) \times \cdots \times \hat{p_{pos}}(u_n \mid g.n)$$

For example, the set $E = \{h(f(a,b)), f(h(b),a)\}$ allows us to define the following conditional probabilities:

$\hat{p_c}(a \mid f.1) = \frac{1}{2}$	$\hat{p_c}(a \mid f.2) = \frac{1}{2}$	$\hat{p_c}(a \mid h.1) = 0$	$\hat{p_r}(a) = 0$
$\hat{p_c}(b \mid f.1) = 0$	$\hat{p_c}(h \mid f.2) = 0$	$\hat{p_c}(b \mid h.1) = \frac{1}{2}$	$\hat{p_r}(h) = \frac{1}{2}$
$\hat{p_c}(h \mid f.1) = \frac{1}{2}$	$\hat{p_c}(b \mid f.2) = \frac{1}{2}$	$\hat{p_c}(h \mid h.1) = 0$	$\hat{p_r}(b) = 0$
$\hat{p_c}(f \mid f.1) = 0$	$\hat{p_c}(f \mid f.2) = 0$	$\hat{p_c}(f \mid h.1) = \frac{1}{2}$	$\hat{p_r}(f) = \frac{1}{2}$

and the probability of the tree $t = h(f(a,b))$ is computed as follows:

$$\hat{p_a}(t) = \hat{p_r}(h) \times \hat{p_c}(f \mid h.1) \times \hat{p_c}(a \mid f.1) \times \hat{p_c}(b \mid f.2) = \left(\tfrac{1}{2}\right)^4 = \tfrac{1}{16}$$

If t is a subtree of a tree of the learning sample (t must be different from the whole tree), then we compute its probability using $\hat{p_{pos}}$ and taking into account

[1] The base case of recursion corresponds to symbols of arity 0

its position relatively to its parent. For example, the probability of the subtree $f(a,b)$ of the set E is computed as follows:

$$\hat{p_a}(f(a,b)) = \hat{p_c}(f \mid h.1) \times \hat{p_c}(a \mid f.1) \times \hat{p_c}(b \mid f.2) = \left(\frac{1}{2}\right)^3 = \frac{1}{8}.$$

2.3 Pruning with Confidence Intervals

In the previous section, we have proposed a way to construct a probability distribution from a set of trees E. We show here how we can use this distribution to *a priori* prune subtrees considered statistically irrelevant for the future learning process. We consider a subset S of $MSub(E)$ and we compute a confidence interval, according to a risk α. We look for an interval $[pmin; 1]$ such that a proportion $1-\alpha$ of subtrees has a probability greater than or equal $pmin$: $p(p_a(t) \geq pmin) = 1-\alpha$.

According to the Central Limit Theorem, the mean $\overline{\hat{p_a}(t_S)}$ of the probabilities of the elements of S follows a normal distribution with an expected value μ and a standard deviation $\frac{\sigma}{\sqrt{|S|}}$. Then the confidence interval around μ is defined as follows: $\mu \in \overline{\hat{p_a}(t_S)} \pm \frac{\hat{\sigma}}{\sqrt{|S|}} \times u_\alpha$, where $\hat{\sigma}$ is the estimated standard deviation and u_α the $(1-\alpha)$-percentile of the normal distribution. Let $pmin$ be the lower bound of this interval, such that $pmin = \overline{\hat{p_a}(t_S)} - \frac{\hat{\sigma}}{\sqrt{|S|}} \times u_\alpha$.

Once this lower bound is computed, our decision rule is to delete all the subtrees of S whose probability is lower than $pmin$. This leads us to define formally the notion of relevance of a subtree. This definition is in relation with the definition of irrelevant features proposed in [11].

Definition 4 *Let a distribution D on a set of subtrees S, modeling the same concept, and let α be a risk. A subtree t is $(1-\alpha)$-relevant in S if and only if $\hat{p_a}(t) \geq \overline{\hat{p_a}(t_S)} - \frac{\hat{\sigma}}{\sqrt{|S|}} \times u_\alpha$.*

To end this section introducing our pruning method, we synthesize the main steps of our approach in Algorithm 1[2] .

3 Partitioning Subtrees with Regular Tree Patterns

In our method we need to compare the probabilities of subtrees of a sample. We think that the comparison of the probabilities of two subtrees that never appeared in the same places (positions) in the sample is irrelevant. Indeed they *a priori* do not model the same concept and then we assume that they are not comparable. We propose in this section a method of partitioning using regular tree patterns, *i.e.* tree patterns with only one variable. This approach ensures that two subtrees that appear in the same context (*i.e.* subtrees with the same ancestors and siblings) will be in the same partition.

Definition 5 *A regular tree pattern is a tree defined on a signature $(S, V \cup \{X\}, \alpha, \sigma)$ where X is a variable and the tree has exactly one leaf labeled by X.*

[2] The probability distribution is adapted to delete subtrees with specific symbols

```
Data: E set of trees
      α real
begin
    Construct the probability distribution with E;
    T ← partitioning MSub(E);
    foreach S ∈ T do
        Compute an interval [pmin; 1] for the probabilities of the subtrees of S
        to the risk α;
        foreach t ∈ S do
            Adapt the probability distribution in deleting the instance of t;
            Delete t if it is not (1 − α)-relevant or if p_a(t) = 0;
            Rebuild the probability distribution in adding the instance of t;
        end
    end
end
```

Algorithm 1: Pruning by confidence interval

Let t a regular tree pattern and t' an ordinary tree. We denote $t._{\#}t'$ the substitution of the variable X of t by the tree t'.

To construct a partition of the multi-set of subtrees, our approach consists in extracting all the regular tree patterns definable from a sample of trees. The set of all regular tree patterns is given by $\{t \mid \exists t' \in MSub(E) \text{ and } t._{\#}t' \in E\}$.

Each pattern t allows us to define a class π_t of the partition of the multi-set of subtrees. All the subtrees which can be concatenated to t to obtain a tree of the learning sample belong to this partition[3]: $\pi_t = \{t' \in MSub(E) \mid t._{\#}t' \in E\}$.

4 Evaluation in the Context of Learning Stochastic Tree Automata

We now present experimental results which justify the interest of our method as a pre-process of the learning task. Since we work on trees, we propose to evaluate our data reduction approach in the framework of learning stochastic tree automata from a sample of trees [8]. In such a context, we have a sample of trees, supposed to be generated from a probability distribution. The objective is to learn the probabilistic model which has generated the data. We propose to compare the automata inferred with noisy data and those induced after the pruning process. We achieved two series of experiments. The first one deals with situations where the target automaton is *a priori* known. In this case, we can use a measure of distance between the inferred model to the target automaton. However, we do not always know this one. Then, we also evaluate our approach in a second series of experiments, using a perplexity measure. This criterion assesses the relevance of the model on a test sample.

[3] The complexity of the approach, relative to the size of trees, depends on the partitioning

4.1 Stochastic Tree Automata

A tree automata [13, 14] defines a regular language on trees as a finite automaton defines a regular language on strings. Stochastic tree automata are an extension of tree automata, defining a probability distribution on the tree language defined by the automaton. We use an extension of these automata taking into account the notion of type: stochastic many-sorted tree automata (SMTA) defined on a signature [15]. We do not detail here these automata and their learning method. The interested reader can refer to [8, 4]. We only specify that a learned stochastic tree automaton allows one to associate a probability to each tree of a sample. Since we apply a pruning method on a set of sorted trees, we have to ensure that pruned trees still respect a signature. Practically, we replace pruned subtree by a unique symbol which does not appear in the sample and has a different type.

4.2 Evaluation Criteria

Distance from the Target Automaton: [16] defines distances between two hidden Markov models introducing the co-emission probability, as the probability that two independent models generate the same string. [17] presents an adaptation of the co-emission to stochastic tree automata. The co-emission probability of two stochastic tree automata $M1$ and $M2$, constructed over the same signature Σ, is denoted $A(M1, M2)$ and defined as follows: $A(M1, M2) = \sum_{t \in \Sigma^T} P_{M_1}(t) * P_{M_2}(t)$. Where $P_{M_i}(t)$ is the probability of t given the model M_i. The co-emission probability allows us to define a distance D_a which can be interpreted as the measure of the angle between the vectors representing automata in a space where the base is the set of trees of Σ^T.

Definition 6 *The distance D_a between two automata $M1$ and $M2$ is defined by:*

$$D_a(M1, M2) = \arccos\left(\frac{A(M1,M2)}{\sqrt{A(M1,M1) * A(M2,M2)}}\right)$$

Perplexity measure In the case of tree automata, the quality of a model A can be evaluated by the average likelihood on a set of trees S relative to the distribution defined by A: $LL = \left(\frac{1}{\|S\|} \sum_{j=1}^{|S|} \log P_A(t_j)\right)$[4]. A perfect model can predict each element of the sample with a probability equal to one, and so $LL = 0$. In a general way we consider the perplexity of the test set which is defined by $PP = 2^{LL}$. A minimal perplexity ($PP = 1$) is reached when the model can predict the probability of each element of the test sample. Therefore we consider that a model is more predictive than another if its perplexity is lower. A problem occurs when a tree of the test sample cannot be recognized by the learned automaton A. Actually the probability of this example is 0 and the perplexity cannot be computed. To avoid this problem a classical method consists in smoothing the distribution of the learned model using an interpolation approach [18] with an unigram model A_0 recognizing all the examples. The probability of a tree t, in our experiments, is then given by:

$$\hat{P}(t) = 0.9 * p(t|A) + 0.1 * p(t|A_0)$$

[4] $\|S\|$ is the number of nodes in S and $P_A(t_j)$ is the probability of t_j according to A

4.3 Experimentations

Recall that our objective is to study the interest of our pruning method in the context of learning tree automata from noisy data. Then we need to be able to artificially corrupt the data. When we work with positive and negative instances, a classic approach to noise data is to invert the labels of some examples. In our framework, all examples are unlabeled (more exactly they are only positive) and we need to introduce the noise in the structure of the examples. Our noise protocol consists in changing a proportion γ of the leaves of the trees. In our experiments we artificially corrupt the training samples with values of γ varying from 1% to 50%, and we apply our pruning method with different values of α from 0.01 to 0.25. We present, at first, the results on the experiments considering a target automaton. Then we present those with a target automaton *a priori* unknown. We use two criteria of performance to evaluate the results: the mean of the quality measure (the distance D_a or the perplexity measure) on all the levels of noise and the standard deviation around this mean. We also test the significance of our results using a Student paired t-test over the means with a critical risk of 5%. Note that all the results presented correspond to the optimal α value.

Experimentations Knowing the Target Automaton We suppose here that the target automaton is *a priori* known. This automaton allows us to generate a sample of trees according to the distribution defined by this automaton. We first infer an automaton from a noisy sample (one for each level of noise). Then we apply our pruning method on the noisy sample with different values of α. We infer an automaton with each pruned sample. We finally measure the distance D_a between each of the inferred automata and the target one. We achieved our experiments on five artificial datasets. One on a tree grammar representing stacks of objects, one on a simple grammar representing conditional statements (denoted Cond.), one on a grammar on boolean expressions (denoted Bool.) and two other artificial datasets (Art1 and Art2). The results are synthesized on Table 1. We give, for each dataset, the size of the initial sample (IS) in the number of subtrees, and the percentage of reduction of our pruning method (Red). In the same table we also give the average distance D_a [5] observed between the target automaton and the inferred automata $\pm$ the standard deviation. $\overline{D_a\ I}$ represents the distance relative to the automaton learned on the initial noisy sample, and $\overline{D_a\ Pr}$ the distance to the one learned from the pruned sample. In order to assess the relevance of our approach, we decided to compare its performances with those obtained by a simple Monte Carlo selection (denoted by MCS in the table). To achieve this comparison, we randomly removed the same proportion of subtrees (i.e. Red %) from the initial multi-set of subtrees ($MSub(E)$). Then, the described results are obtained from learning sets with exactly the same size. The last two columns concern the significance of the results (**Sig1** for Pr vs I and **Sig2** for Pr vs MCS). The experiments have shown that, not only the pruning of the noisy sample allows us to learn an automaton closer to the target one with

[5] The means computed are the means of the distances on the different levels of noise

Base	IS	Red	$\overline{D_a\ I}$	$\overline{D_a\ Pr}$	$\overline{D_a\ MCS}$	Sig1	Sig2
Stacks	35724	5%	0.721 ± 0.514	0.420 ± 0.332	0.869 ± 0.496	yes	yes
Cond.	71863	5%	0.618 ± 0.187	0.590 ± 0.224	1.050 ± 0.316	no	yes
Bool.	43185	26%	0.350 ± 0.240	0.242 ± 0.230	0.227 ± 0.142	yes	no
Art1	33137	0.5%	1.230 ± 0.265	0.556 ± 0.237	1.374 ± 0.339	yes	yes
Art2	30113	7.4%	0.846 ± 0.398	0.599 ± 0.247	1.011 ± 0.485	yes	yes

Table 1. Distances D_a to the target automaton

an average reduction of 8.8% of subtrees, but also our approach is highly better than the Monte Carlo sampling. The obtained results are particularly interesting on the databases Stacks, Bool., Art1 and Art2. On these databases, the standard deviation of computed distances without pruning is significantly higher than the one of computed distances with pruning.

Experimentations without Knowing the Target Automaton In this context, we consider to have a characteristic sample of trees. We divide this sample in two sets: a training set and a test set. As we want to evaluate our approach in the context of noise, we corrupt the training set with the different levels of noise. We learn an automaton for each of the noisy samples. Then we apply our pruning method on each of these samples with the different values of α. We learn a tree automaton for each of the pruned sample obtained. Then we evaluate all the inferred automata on the test set with the perplexity measure. Note that the test set is never noised. To make our experiments we use a 5-folds cross-validation. We did our experiments on 8 datasets. We have used a sample of each of the five artificial databases presented in the previous section. We also evaluate our method on real data with a sample of the PKDD'02 discovery challenge[6] database (dataset on hepatitis) obtained as described in [4]. We also use the database Student Loan of the UCI Irvine [19] and a dataset on the toxicity of the tacrine molecule presented in [20]. These two datasets correspond to structured data in 1st order logic and were converted into trees according to the principle presented in [15]. The results are presented on Table 2. We give for each dataset the average percentage of sample reduction with the pruning method (Red). The average perplexity measure with noisy data without pruning ($\overline{P\ I}$) and the average perplexity obtained after the pruning process ($\overline{P\ Pr}$) ± the standard deviation. The last column indicates if the results are significant. The results have shown that the perplexity is reduced in most of the cases, except for the Stacks and the Tacrine, with an average reduction of 10% of the training sample. In most of the experiments, the standard deviation is lower after a pruning phase than with no pruning. This remark tends to confirm that our method is relatively robust.

We summarize our results in a concise way on Figure 1. Each dot represents a database. A dot under the bisecting line expresses the fact that the pruning approach is better than the one with no pruning.

[6] http://lisp.vse.cz/challenge/ecmlpkdd2002/

Base	Red	$\overline{P\ I}$	$\overline{P\ Pr}$	Significant
Stacks	6%	2.23 ±0.274	2.24 ± 0.279	no
Cond.	4.6%	1.73 ± 0.241	1.70 ± 0.202	no
Bool.	13%	3.49 ± 0.302	3.47 ± 0.270	yes
Art1	17%	3.61 ± 0.219	3.26 ± 0.210	yes
Art2	3%	4.23 ± 1.456	3.96 ± 0.837	no
PKDD02	0.5%	14.84 ± 7.498	13.52 ± 7.235	yes
Tacrine	12%	5.78 ± 4.171	6.01 ± 4.111	no
Student Loan	17%	11.9 ± 1.871	11.43 ± 2.112	yes

Table 2. Results with the perplexity measure

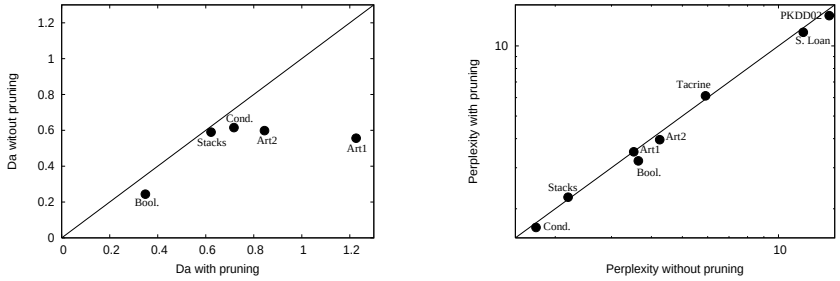

Fig. 1. Results of the experimentations

5 Conclusion

In this paper we have presented an original approach allowing to prune subtrees in a set of trees. This approach of data reduction can be considered as an hybrid approach. It can actually delete whole trees and is related to prototype selection. It also allows one to delete some parts of the examples and the method is close to feature selection, with the difference that a subtree is deleted locally for one example. Our pruning method is based on the utilization of confidence intervals computed from a probability distribution on subtrees appearing in a same context in the examples. Our experiments have shown that our approach is robust and allows us to learn more efficient models closer to the target one in terms of distance or with a smaller perplexity. Moreover, our method allows us to reduce the size of the learning sample, this can be crucial in some applications.

Since we worked in the framework of the inference of probabilistic models on trees, we have to avoid the deletion of too many subtrees. Indeed, in this context, we aim at estimating correctly the distribution of the data. If we delete too many subtrees, we modify considerably the distribution of the initial learning sample and then it may be almost impossible to infer a correct distribution. A simple solution can be to tune the parameter α. Another complementary point of view consists in working on the method of partitioning. A larger partitioning of the learning subtrees, *i.e.* with a high number of classes, may more often lead to delete fewer subtrees than a smaller one. Our approach was experimentally evaluated as pre-process of a learning task. Since its works on any database of trees, and because a production rule can be seen as a tree, we are currently working on the simplification of knowledge bases.

References

1. De Raedt, L.: Data mining in multi-relational databases. In: 4th European Conference on Principles and Practice of Knowledge. (2000) Invited talk.
2. Zaki, M.: Efficiently mining frequent trees in a forest. In: Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (KDD). (2002)
3. Termier, A., Rousset, M., Sebag, M.: Treefinder: a first step towards xml data mining. In: International Conference on Data Mining ICDM02. (2002)
4. Habrard, A., Bernard, M., Jacquenet, F.: Generalized stochastic tree automata for multi-relational data mining. In: 6th International Colloquium on Grammatical Inference. ICGI 2002. Volume 2484 of LNAI., Amsterdam, Springer (2002) 120–133
5. Miyahara, T., Suzuki, Y., Shoudai, T., Uchida, T., Takahashi, K., Ueda, H.: Discovery of frequent tag tree patterns in semistructured web documents. In: PAKDD 2002, Taipei, Taiwan (2002)
6. Goldman, S.A., Kwek, S.S.: On learning unions of pattern languages and tree patterns. In: 10th Algorithmic Learning Theory conference. Volume 1720 of LNAI. (1999) 347–363
7. Amoth, T.R., Cull, P., Tadepalli, P.: On exact learning of unordered tree patterns. Machine Learning **44** (2001) 211–243
8. Carrasco, R., Oncina, J., Calera-Rubio, J.: Stochastic Inference of Regular Tree Languages. Machine Learning **44** (2001) 185–197
9. Rico-Juan, J., Calera-Rubio, J., Carrasco, R.: Stochastic k-testable tree languages and applications. In: ICGI 2002. Volume 2484 of LNAI., Amsterdam (Nederland), Springer-Verlag (2002) 199–212
10. Wilson, D., Martinez, T.: Reduction techniques for instance-based learning algorithms. Machine Learning **38** (2000) 257–286
11. John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: 11th International Conference on Machine Learning. (1994) 121–129
12. Brown, P., Pietra, V.D., deSouza, P., Lai, J., Mercer, R.: Class-based n-gram models of natural language. Computational Linguistics **18** (1992) 467–479
13. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
14. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications . Available on: `http://www.grappa.univ-lille3.fr/tata` (1997)
15. Bernard, M., Habrard, A.: Learning stochastic logic programs. In Rouveirol, C., Sebag, M., eds.: Work-in-Progress Track at the 11th International Conference on Inductive Logic Programming. (2001) 19–26
16. Lyngsø, R., Pedersen, C., Nielsen, H.: Metrics and similarity measures for hidden Markov models. In: 7th International Conference on Intelligent Systems for Molecular Biology, ISMB '99 Proceedings, Heidelberg, Germany, AAAI Press USA (1999) 178–186
17. Carrasco, R.C., Rico-Juan, J.R.: A similarity between probabilistic tree languages: application to xml document families. Pattern Recognition, in press (2002)
18. Ney, H., Essen, U., Kneser, R.: On structuring probabilistic dependences in stochastic language modelling. Computer Speech and Language **8** (1994) 1–38
19. Blake, C., Merz, C.: University of California Irvine repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/` (1998)
20. King, R., Srinivasan, A., Sternberg, M.: Relating chemical activity to structure: An examination of ILP successes. New Generation Computing, Special issue on Inductive Logic Programming **13** (1995) 411–434