Inference of Finite Automata: Reducing the Search Space with an Ordering of Pairs of States

François Coste and Jacques Nicolas

IRISA- INRIA
Campus de Beaulieu, F-35042 Cedex, France

Abstract. We investigate the set of all minimal deterministic finite automata accepting a given set of words and rejecting another given set of words. We present several criteria to order the exploration of the corresponding search space. Three criteria are shown to have a very good behavior with respect to the pruning they imply in the search space. Best results have been obtained for the prefix ordering. We have also worked on a new dynamic ordering based on an entropy computation.

keywords :grammatical inference, DFA, constraints, search tree, entropy.

1 Introduction : regular grammar inference

Regular Grammar Inference is an important inductive inference issue with applications in language processing, pattern recognition and genomics. It is defined as the process of learning a regular language from examples of words of this language and words that do not belong to the language. More precisely, we are looking in this paper for all minimal deterministic finite automata accepting a given set of words and rejecting another given set of words. Formally, we are concerned with the following DFA learning problem:

Given a set of positive sentences I_+ and a set of negative sentences I_-, find the set BS_d (Border Set, Dupont& al. 1994, of deterministic automata) of all automata A verifying:

1. A is a deterministic finite state automata ;
2. I_+ is structurally complete for A (i. e. there exists an acceptance of I_+ such that every transition of A is exercised and every final state of A is used as an accepting state);
3. $L(A)$, the language accepted by A, does not contain any string of I_- ;
4. $L(A)$ is a most general language
   (i.e. no other solution A' is such that $L(A) \subseteq L(A')$).

The natural generality relation between languages is inclusion, corresponding to inclusion between sets of accepted words. In case of a representation of languages with automata, a weaker generality relation is known: the set of finite automata may be partially ordered with a derivation relation, corresponding to the merging of states in the automata:

Definition 1. (Derived automata A/π) Given an automata $A = (Q, \Sigma, \delta, q_0, F)$ and a partition $\pi = (B_0, B_1, \ldots, B_r)$ of the set of states Q, the derived or quotient automaton $A/\pi = (\pi, \Sigma, \Delta, B_0, R)$ is defined as follows:

- initial sate : $q_0 \in B_0$;
- final states : $R = \{B_i \in \pi, \exists q \in B_i \, tq \, q \in F\}$;
- transition function : $B_j \in \Delta(B_i, a)$ iff $\exists q \in B_i, q' \in B_j$ such that $q' \in \delta(q, a)$.

The set of all automata derived from A is a lattice denoted $Lat(A)$.

In our case, learning may be reduced to a search in the lattice $Lat(PTA(I_+))$ (Dupont& al. 1994), where $PTA(I_+)$ is a most specific automaton specified by I_+, the *prefix tree acceptor*. $PTA(I_+)$ accepts every word of I_+ and no other.

A number of algorithms have been proposed to explore this lattice. Almost all of them are looking for one particular solution. The main algorithms search a solution in the lattice proceeding either with a greedy state merging approach (Trakhenbrot & Barzdin 1973, RPNI Oncina & Garcia 1992, Lang 1992), a beam search strategy (BRIG), or with an optimization method (GIG). Few authors have studied the characterization of the set of all solutions. Ensuring a complete search presents however many advantages : it allows to design an incremental approach, to be more robust with respect to noise in the training set and to select useful training words. Miclet has proposed a heuristic algorithm. An incremental approach using membership queries has been proposed by Parekh and Honavar. Preliminary studies have also been tried in the context free case by Vanlehn and Ball and by Giordano.

In Coste & Nicolas 1997b, we have proposed a compact representation of the set of all solutions, based on a *constraints system*. This constraints system specifies the set F_{Nok} of couples of states that cannot be merged. We have given an efficient algorithm to build the minimal elements of BS_d, reducing the problem to the *coloring of the graph of constraints*. Pairs of states may be considered as binary attributes taking value $=$ or $\neq$, depending on whether states are merged or not. In these experiments, we have seen that the efficiency of the search is very sensitive to the order in which attributes are considered.

In algorithms such as RPNI (Oncina & Garcia 1992) or the algorithm of Lang (Lang 1992), the states to be merged are chosen in a breadth-first predetermined order (prefix order): a state is ordered with respect to the word leading to this state. A recent study (Lang 1997) shows that better results can be obtained by considering candidate merges in order of the amount of evidence supporting them, as claimed by De la Higuera, Oncina and Vidal . Basically, Lang proposes to merge states with most matching labels in the suffix trees associated with the candidate states. In the following we note Rlb the order resulting from this criterion.

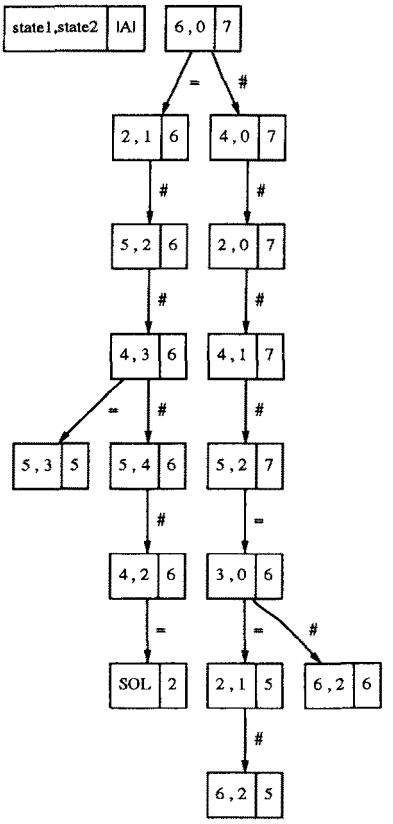
We propose to study the influence of these orders on the search space while looking for all solutions. It is thus interesting to try new criteria for the selection of attributes. Using the same idea of dynamically computing the evidence of a merge we have developed an entropy based criterion. We present the search tree and this criterion in the next section. Experiments comparing orders are reported in the last section.

2 Ordering the choices of state mergings

2.1 The search space as a search tree

We have seen that the search space is a set of partitions and that there is one operation, namely merging two blocks in a partition, to move from one node to another more general one in the search space.

An alternative representation of this space consider each pair of states to be an attribute with two possible values : $=$ if states have to be merged and $\neq$ if states differ in the target automaton. At each node of the search tree, one may associate a set of automata (corresponding to an incompletely specified, current automaton) all of them derived from the PTA, where all merges along the path from the root to the given node have been made, all states that cannot be merged along this path are indeed different, and all consequences of these choices have been drawn (i.e. propagation of constraints has been done).



We give a search tree of sample 1 of language 2, for a random selection of attributes.

It is possible to obtain a complete search tree if one follows every possible choice along the branches of this tree, until either a single solution or a failure is discovered. In this way, heuristics of every known algorithm searching a single solution may be used for the selection of attributes in this tree and thus these algorithms may be extended naturally to the search of all solutions. This is possible because, unlike other algorithms, we explicitly and dynamically manage the F_{Nok} set of incompatible mergings.

The question is then to find the most efficient heuristic with respect to this new goal, and this is not a trivial task since a given algorithm may well find very quickly a first solution, taking advantage of a property of this solution, and then be laborious in finding the other ones, that do not respect this property.

2.2 The search tree as a decision tree

We have so far presented the search space as an ordered set of choices of pairs of states. Another way to look at it is that the search tree is a kind of decision tree, the leaves of this tree being labeled either success, failure or incomplete, depending on whether the corresponding automata are either correct, rejected or incompletely specified.

Given this framework, reducing the amount of search to be done is equivalent to finding the smallest tree where all leaves contain either a single solution automaton or lead to a failure. Following the TDIDT methodology (Quinlan 1986), we have tried various criteria in order to select the best attribute at each node.

However, one has to keep in mind that this is an analogy and not the usual framework, since we do not know how many solutions or failures are below a given node. All we can do is to estimate this number. Defining this estimate for each node in the decision tree is the purpose of the next part of this section.

Estimating the number of solutions In order to estimate the number of solutions, we split the set of states into two parts. The first part, *the pseudo-clique* (PC), corresponds to a set of candidates such that every other state (in the second part denoted $\overline{PC}$) is supposed to be merged with one of them to obtain the target automaton. Name "pseudo-clique" refers to graph F_{Nok}, where states in PC are likely to be linked and form a clique (or a densest 'almost' clique). We then only consider the bipartite subgraph of F_{Nok} between PC and $\overline{PC}$. Once a pseudo-clique is chosen, we estimate the number of solutions by considering possible completions of this subgraph (each one specifying an automaton). Let m be the number of states of the target automaton, $|A|$ be the number of states of the current automaton, $degree(x)$ denotes the number of states of PC related to x in graph F_{Nok}. States of PC are denoted p_i and states of $\overline{PC}$ are denoted q_i.
The total number of possible graphs tot is:

$$tot = \prod_{1 \leq i \leq |A|-m} 2^{(m-degree(q_i))}$$

Then, we can roughly estimate the number of solution graphs p as the number of possible mergings of states in $\overline{PC}$ with states in PC.

$$p = \prod_{1 \leq i \leq |A|-m} (m - degree(q_i))$$

The number of failure configurations n is the number of graphs where a state of $\overline{PC}$ cannot be merged with any state of PC.

$$n = \sum_{1 \leq i \leq |A|-m} \left(\prod_{1 \leq j < i} (2^{m-degree(q_j)} - 1) \cdot \prod_{i < j \leq |A|-m} 2^{m-degree(q_j)} \right)$$

Therefore, the number i of incomplete configurations is

$$i = tot - (p + n)$$

Then the estimated entropy of a given node is

$$I = -\frac{p}{tot} log_2(\frac{p}{tot}) - \frac{n}{tot} log_2(\frac{n}{tot}) - \frac{i}{tot} log_2(\frac{i}{tot})$$

Finally, averaging entropies of the *left* and *right* edges of a given node, we propose to choose the attribute (i.e. the pair of states) minimizing E:

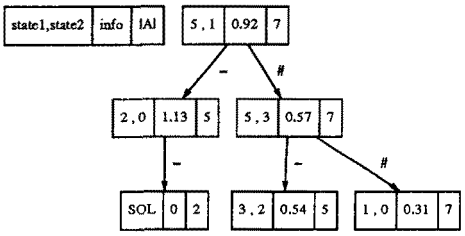
$$E = tot_{left} I_{left} + tot_{right} I_{right}$$

3 Experimentation

3.1 Setting

We have used a benchmark described in Dupont 1996 for a first validation of ideas developed in this paper. Target automata are small (< 6 states) but structurally complex. For instance, L_{11} is the language accepting an even number of a and an odd number of b.
For each language, 20 training samples have been drawn. The size of these sets of words is less than or equal to 50. The size of words varies from 1 to 40.

We have used four orderings while building a complete search tree of the solutions: Random order r (20 trials have been made for each training set); Prefix order p; Entropy-based order e and Rlb order R (Lang 1997). In each case, pseudo-cliques have been computed and used to detect failure nodes.



(if a clique of size greater than m is detected, it means that the corresponding automaton is not minimal, and the search may be pruned). As an illustration, we give the search tree of sample 1 of language 2 for the entropy criterion.

3.2 Results

Lang.	#s/#t	#sol	APTA	#FN	#nr	#np	#ne	#nR
L1	1/1	1	3.2	0	2.5	2.0	2.0	2.0
L2	2/2	1	25.3	43.0	10.3	2.9	3.0	3.1
L3	4/7	1.75	180.9	998.2	4325.6	13.7	16.1	52.9
L4	3/5	8.5	102.2	192.2	3297.2	58.7	71.6	268.8
L5	4/8	2.1	65.1	217.8	30836.7	32.3	48.5	73.1
L6	3/6	2.45	40.7	108.5	9842.9	19.3	18.7	73.4
L7	4/7	4.75	85.2	545	27291.4	92.6	119.9	405.5
L8	2/2	1.25	19.0	43.0	3.6	2.7	2.7	3.1
L9	4/7	1.65	60.8	357.9	114.7	5.8	7.0	7.1
L10	5/6	2.4	81.2	431.7	97.5	15.7	10.4	16.7
L11	4/8	2.85	64.8	214.85	37035.1	27.0	55.6	74.2
L12	3/3	1.5	37.6	120.2	8.6	4.4	4.1	4.2
L13	2/4	1.25	32.4	70.7	475.8	5.8	5.6	5.6
L14	3/4	1.4	37.6	100.5	30.1	5.5	6.2	6.4
L15	4/6	2.1	70.7	353.3	35.6	8.9	9.1	9.2

#s/#t characterizes the size of target automata in terms of number of states and transitions, #sol is the mean number of minimal automata in BS_d. #FN is the mean size of F_{Nok} at the root node. #nr, #np, #ne, #nR are the mean number of nodes of the tree for respectively a random, a prefix, an entropy, and a Rlb selection.

Results first show how important is the choice of a good ordering for the reduction of the search space. A random selection of nodes may increase the

size of the tree by several orders of magnitude, with respect to best criteria. Second, results are in favor of the prefix order. The refined Rlb order is not only not necessary but give worse results in almost every case. The entropy order behaves well but seems to be not as efficient as the prefix order. We have to temperate these observations with two important remarks : our benchmark contains target automata of small size and furthermore, each learning sample admits a very low number of solutions. A second remark is that it remains a full range of possible variations to improve the calculation and use of the entropy criterion that we have not checked.

4 Conclusion

We have presented a search space for regular grammatical inference and experimented with various ordering criteria allowing a substantial reduction of the amount of search needed in a complete strategy. The benchmark needs of course to be completed with more difficult problems, increasing the size of the vocabulary and the size of the target automaton.

The first important result is that ordering the choices of pairs of states to be merged remains a key issue when considering the "all solutions" search problem. The simple prefix order seems to be a good candidate for this search. We have studied a new entropy based criterion that behaves well either but is more costly. We expect it to be interesting in large search spaces with a great number of solutions. Finally, splitting the set of states of an automaton into two parts, based on the number of impossible mergings between them, and taking advantage of this partition to further prune the search space has contributed to very small search trees in our experiment. This point has to be emphasized since algorithms inferring automata usually consider possible mergings but not impossible ones.

References

Coste, F. and Nicolas, J. : Regular Inference as a Graph Coloring Problem. Workshop on Grammar Inference, Automata Induction, and Language Acquisition (ICML' 97), Nashville, TN.

Dupont, P.; Miclet, L.; and E.Vidal. What is the search space of the regular inference ? *ICGI'94, Grammatical inference and Applications* 25–37. Springer Verlag.

Dupont, P. *Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue.* Ph.D. Dissertation, Ecole Nationale Supérieure des Télécommunications.

Lang, K. Random DFA's can be Approximately Learned from Sparse Uniform Examples. In *proceedings of the fifth annual ACM Workshop on Computational Learning Theory* 45–52, July 1992.

Lang, K. Merge Order count NECI Tech Report, Sept26, 1997

Oncina, J., and Garcia, P. Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis* 49 – 61.

Quinlan, J.R. Induction of decision trees. *Machine Learning* (1):81–106.

Trakhenbrot, B. and Barzdin, Y., Finite Automata : Behavior and Synthesis *Amsterdam, North Holland Pub. Comp,*