Learning Limited Context Restarting Automata by Genetic Algorithms* (Technical report)

Stanislav Basovník and František Mráz

Charles University, Faculty of Mathematics and Physics
Department of Computer Science, Malostranské nám. 25
118 00 PRAHA 1, Czech Republic
sbasovnik@gmail.com, mraz@ksvi.ms.mff.cuni.cz

Abstract. We propose a genetic algorithm for learning restricted variants of restarting automata from positive and negative samples. Experiments comparing the proposed genetic algorithm to algorithms RPNI and LARS on sample languages indicate that the new algorithm is able to infer a target language even from a small set of samples.

1 Introduction

Restarting automata [1] were introduced as a model for linguistically motivated method of checking correctness of sentences of a natural language by stepwise simplification of the input sentence while preserving its (non)correctness.

A restarting automaton can be represented as a finite set of meta-instructions defining possible reductions of a current word. In a general restarting automaton, each reduction consists in rewriting a short subword by even shorter word. The possibility to apply a meta-instruction is controlled by the content of the whole tape to the left and to the right from the place of rewriting. The left and right context must belong to given regular languages which are comprised by the meta-instruction as regular constraints.

Several variants of restarting automata were studied in numerous papers [1]. In this paper we propose more restricted variant of restarting automata for which the possibility to apply a meta-instruction is controlled by a fixed finite size context around the rewritten subword – restarting automata with limited context (lc-R-automata). We propose a special version of a genetic algorithm to learn lc-R-automata from both positive and negative samples. Due to their simpler definition, the learned lc-R-automata are much easier to interpret by humans than general restarting automata.

In Section 2 we introduce lc-R-automata and their several more restricted variants. Section 3 presents the proposed genetic algorithm for learning lc-R-automata from positive and negative samples. Then we compare the proposed algorithm with well-known learning algorithms – RPNI [2] and LARS [3].

* This work was partially supported by the Grant Agency of Charles University in Prague under Grant.-No. 120709/MFF/A-INF and by the Grant Agency of the Czech Republic under Grant-No. P103/10/0783 and P202/10/1333.

2 Definitions and Notations

Let λ denote the empty word and $|w|$ denote the *length* of the word w.

Definition 1. *A* limited context restarting automaton (lc-R-automaton) *is a system $M = (\Sigma, \Gamma, I)$, where Σ is an input alphabet, Γ is a working alphabet containing Σ, and I is a finite set of* meta-instructions *of the following form $(\ell \mid x \to y \mid r)$, where $x, y \in \Gamma^*$ such that $|x| > |y|$, $l \in \{\lambda, ¢\} \cdot \Sigma^*$ and $r \in \Sigma^* \cdot \{\lambda, \$\}$.*

A lc-R-automaton $M = (\Sigma, \Gamma, I)$ induces a reduction relation $\vdash_M$ as follows: for each $u, v \in \Gamma^*$, $u \vdash_M v$ if there exist words $u_1, u_2 \in \Gamma^*$ and a meta-instruction $(\ell \mid x \to y \mid r)$ in I such that $u = u_1 x u_2$, $v = u_1 y u_2$, ℓ is a suffix of $¢u_1$ and r is a prefix of $u_2\$$. Let $\vdash_M^*$ denote the reflexive and transitive closure of $\vdash_M$. The language accepted by lc-R-automaton M is $L(M) = \{w \in \Sigma^* \mid w \vdash^* \lambda\}$.

A lc-R-automaton M accepts exactly the set of input words which can be reduced to λ. Obviously, λ is in $L(M)$, for each lc-R-automaton M.

Example 1. Let $M = (\{a, b\}, \{a, b\}, I)$, where $I = \{(a \mid abb \to \lambda \mid b), (¢ \mid abb \to \lambda \mid \$)\}$, be a lc-R-automaton. Then $aaabbbbbb \vdash_M^c aabbbb \vdash_M^c abb \vdash_M^c \lambda$ and the word a^3b^6 belongs to $L(M)$. It is easy to see that $L(M) = \{a^n b^{2n} \mid n \geq 0\}$.

We consider several even more restricted variants of lc-R-automata. We say, that R is of type (in the following $x, y \in \Gamma^*$, $u \in \{\lambda, ¢\} \cdot \Gamma^*$, $v \in \Gamma^* \cdot \{\lambda, \$\}$):

$\mathcal{R}_0$ if I is arbitrary finite set of rules without any restriction.
$\mathcal{R}_1$ if I contains only rules of the following form $(u|x \to \lambda|v)$ or $(u|x \to a|v)$, where $a \in \Gamma$.
$\mathcal{R}_2$ if I contains only rules of the following form $(\{\lambda, ¢\}|x \to \lambda|\{\lambda, \$\})$ or $(\{\lambda, ¢\}|x \to a|\{\lambda, \$\})$, where $a \in \Gamma$.
$\mathcal{R}_3$ if I contains only rules of the following form $(\{\lambda, ¢\}|x \to \lambda|\$)$ or $(\{\lambda, ¢\}|x \to a|\$)$, where $a \in \Gamma$.

Basovník in [4] studied the power of lc-R-automata. lc-R-automata of type $\mathcal{R}_0$ recognize exactly the class of growing context-sensitive languages. lc-R-automata of type $\mathcal{R}_2$ recognize exactly the class of context-free languages and lc-R-automata of type $\mathcal{R}_3$ recognize exactly the class of regular languages.

The problem of inferring a lc-R-automaton for a target language $L \subseteq \Sigma^*$ consists in learning its set of rewriting meta-instructions. In this work, lc-R-automata are inferred from input set of positive and negative samples $\langle S_+, S_- \rangle$, where $S_+ \subseteq L$, $S_- \subseteq \Sigma^* \setminus L$, using genetic algorithm [5]. Every individual in a population is a set of rewriting rules. Its fitness is

$$F = 100 \left(1 - \frac{1}{2}\left(\frac{|E_+| - \frac{b}{3}}{|S_+|} + \frac{|E_-|}{|S_-|}\right)\right) \ ,$$

where E_+ is a set of rejected positive samples, E_- is a set of accepted negative samples and b is a bonus for partially reduced rejected positive samples:

$$b = \sum_{s \in E_+} \frac{|s| - |s'|}{|s|} \ ,$$

where s' is the shortest word, that can be created from s using rewriting rules of the evaluated individual.

The genetic algorithm uses three operators:

1. Selection: a tournament selection with elitism – 3 fittest individuals are copied into the new generation without any change; in a tournament two individuals are randomly selected and with probability 0.75 the individual with higher fitness is used for constructing the new generation.
2. Crossover: a random shuffle of rules of two individuals – if some individual is left without rules, a random rule is added to it.
3. Mutation: one of the following changes in an individual is made:
   - with probability 0.1 a randomly chosen rule is deleted,
   - with probability 0.35 a random rule is added,
   - with probability 0.55 a randomly chosen rule is edited:
     - with probability 0.1 a random symbol is deleted,
     - with probability 0.45 a new symbol is randomly inserted,
     - with probability 0.45 a randomly chosen symbol is rewritten.

An initial population contains individuals with single rule. All the rules used in an initial population are of the form $(¢, w, \$)$, where w is one of the positive training samples of minimal length.

3 Results

We have compared learning of lc-R-automata using GAs with two well-known methods for grammatical inference – regular positive and negative inference (RPNI; [2]) for inferring regular languages and learning algorithm for rewriting systems (LARS; [3]) for inferring languages represented by string rewriting systems, because restarting automata can be interpreted as a regulated string rewriting systems, too. The algorithms were compared on two sets of languages: 15 regular languages used by Dupont in [6] and the following 7 context-free languages: $\{a^n b^n \mid n \geq 0\}$, $\{a^n c b^n \mid n \geq 0\}$, $\{a^n b^{2n} \mid n \geq 0\}$, $\{ww^R \mid w \in \{a,b\}^*\}$, $\{w \mid w = w^R, w \in \{a,b\}^*\}$, the Dyck language of matching parentheses (and), and another Dyck language of matching parentheses of two types (,), [,].

After thorough experiments, we fixed parameters of the genetic algorithm: population size 200, epoch count 400, mutation rate 0.2, crossover rate 0.6, no auxiliary symbols we allowed (working alphabet coincided with the input alphabet). For each tested language L, we generated two sets S_+, S_- of positive and negative samples of size $N = |S_+| = |S_-|$.

Positive samples were generated by breadth-first search of derivations according to a fixed grammar for L, where each generated word was inserted into S_+ with a probability $\frac{i}{10 \cdot N} + 0.5$, until we obtained N positive samples. While negative samples were generated randomly.

For each tested language there were generated 100 sets of positive and negative samples. Each set was randomly split into (i) training samples containing

Table 1. Mean values of the fitness function on test sets achieved by the tested algorithms together with their standard deviations.

	regular languages		context-free languages
	$t_n = 100$	$t_n = 10$	$t_n = 10$
RPNI	99.6 ± 1.2	84.9 ± 18.4	75.3 ± 18.6
LARS	97.3 ± 9.1	80.8 ± 19.8	72.3 ± 20.6
lc-R	98.2 ± 4.5	91.8 ± 10.6	92.6 ± 12.3

t_n positive and t_n negative samples and (ii) testing samples containing t_t positive and t_t negative samples for the learned language ($N = t_n + t_t$). In all our experiments $t_t = 50$.

Learning of restarting automata was performed for all restrictive types (from $\mathcal{R}_0$ to $\mathcal{R}_3$) and the best result was used. In the first experiment we compared all three methods on large training sets of samples ($t_n = 100$) of regular languages. The best method was RPNI, but the learning restarting automata had similar result and for some languages it was even better than RPNI algorithm. During the second and the third experiment the learning methods were tested on a small training sets of samples ($t_n = 10$) from the regular languages and from the context-free languages, respectively. The learning of restarting automata was significantly the best method in these experiments. We can say, that the developed learning method is well-generalizing, particularly for non-regular languages.

Beside nondeterministic lc-R-automata we have proposed their deterministic version in which the meta-instructions are applied in fixed order and on the leftmost position only. Such automata performed similarly as the nondeterministic lc-R-automata but in a fraction of computation time. Relating the lc-R-automata with 'forced determinism' to the Chomsky hierarchy is still ongoing.

References

1. Otto, F.: Restarting automata. In Ésik, Z., Martín-Vide, C., Mitrana, V., eds.: Recent Advances in Formal Languages and Applications. Vol. 25 of Studies in Computational Intelligence. Springer, Berlin (2006) 269–303
2. Oncina, J., García, P.: Inferring regular languages in polynomial update time. In de la Blanca, N.P., Sanfeliu, A., Vidal, E., eds.: Pattern Recognition and Image Analysis. Vol. 1 of Series in Machine Perception and Artificial Intelligence. World Scientific, Singapore (1992) 49–61
3. Eyraud, R., de la Higuera, C., Janodet, J.C.: Representing languages by learnable rewriting systems. In Paliouras, G., Sakakibara, Y., eds.: ICGI 2004. Vol. 3264 of LNAI. Springer, Berlin (2004) 139–150
4. Basovník, S.: Learning restricted restarting automata using genetic algorithm. Master thesis, Charles University, Faculty of Mathematics and Physics, Prague (2010)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Mass. (1989)
6. Dupont, P.: Regular grammatical inference from positive and negative samples by genetic search: The GIG method. In: ICGI '94. Vol. 862 of LNAI. Springer, Berlin (1994) 236–245