МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2002
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2002
Proceedings of Thirty First Spring Conference of
the Union of Bulgarian Mathematicians
Borovets, April 3–6, 2002

LEARNING CONTEXT-FREE GRAMMARS USING INDUCTIVE LOGIC PROGRAMMING METHODS*

Alexander Todorov Grigorov

In this paper we present a method of machine learning of context-free grammars (CFG) using methods adapted from Inductive Logic Programming (ILP). We have modified the inference rules used by the inverse resolution method and applied them for learning grammars. The paper describes the problem, the inference rules and the learning process.

1. Introduction. Inductive logic programming (ILP) is a new rapidly developing field in machine learning [3, 4]. Recent results show that it can be successfully applied for learning tasks in computational linguistics and natural language processing [1, 4]. The system FOIDL [2] shows better results than other machine learning programs in learning the past tense of English verbs. ILP has been successfully used to learn grammar and semantics in the system CHILL [5] to parse natural language queries into logical form.

In this paper we present another approach in applying ILP methods to learn context-free grammars (CFG). The next sections describe the problem, the inference rules and the learning process.

2. The Problem. Here we will formulate the problem of learning context-free grammars. The grammar will expressed in Definite Clause Grammar (DCG) notation. The grammar rules consists of terminal symbols (written in square brackets) and non-terminal symbols. Since we restrict only to context-free grammars non-terminal symbols are written as atoms (i. e. they do not have any arguments). The grammar rules have the form:

$NT \rightarrow S_1, S_2, \ldots, S_n$

where NT is a non-terminal symbol and S_i are terminal or non-terminal symbols.

As input we have a training set of positive examples, background knowledge and a start non-terminal symbol. The training set of positive examples consists of phrases (lists of terminal symbols) that should be recognized by the learned grammar using the start non-terminal. The background knowledge includes all the dictionary grammar rules and a set of some additional grammar rules. The set of additional rules can be an empty one, can be hand written or can be generated by another learning process. The output of the learning process are CFG rules covering the training set examples.

*This research is partially sponsored by the Bulgarian Ministry of Education and Science under Contract No I-813.

Let us consider the following example. Let the training example set consist of the following phrases (sentences):

[a, boy, sings]

[the, boy, sings, a , song]

[the, boy, likes, soccer]

The background knowledge should include the dictionary rules that could look like the following:

(1) $n \rightarrow [boy]$.
(2) $n \rightarrow [song]$.
(3) $n \rightarrow [soccer]$.
(4) $det \rightarrow [a]$.
(5) $det \rightarrow [an]$.
(6) $det \rightarrow [the]$.
(7) $v \rightarrow [sings]$.
(8) $v \rightarrow [likes]$.

It can also include additional grammar rules, for example defining noun phrase:

(9) $np \rightarrow n$.
(10) $np \rightarrow det, n$.

The start symbol is s (sentence) and the problem is to induce CFG rules that can be used to parse correctly the sentences in the training set.

3. The Inference Rules. One of the methods used in Inductive Logic Programming is the inverse resolution. It consists of four inductive inference rules – absorption, identification, intra-construction and inter-construction [3]. These rules are applied to Horn clauses to produce new clauses. There is a great similarity between the structure of Horn clauses and the structure of context-free grammar rules, so such inference rules could be used to learn grammar rules. But also there are some differences – for example the order of the atoms in a Horn clause is *not significant* since conjunction is a symmetric logical operation, while the order of the constituents in a CFG rule *does matter* since this is the way the word order of the language is expressed. We have modified the inductive inference rule of inverse resolution in respect with these differences. The modified inference rules are as follows:

Absorption:

$$\frac{q \rightarrow A \qquad p \rightarrow L^*,\ A,\ R^*}{q \rightarrow A \qquad p \rightarrow L^*,\ q,\ R^*}$$

Identification:

$$\frac{p \rightarrow L^*,\ A,\ R^* \qquad p \rightarrow L^*,\ q,\ R^*}{q \rightarrow A \qquad p \rightarrow L^*,\ q,\ R^*}$$

Intra-construction:

$$\frac{p \rightarrow L^*,\ A,\ R^* \qquad p \rightarrow L^*,\ B,\ R^*}{q \rightarrow A \qquad p \rightarrow L^*,\ q,\ R^* \qquad q \rightarrow B}$$

Inter-construction:

$$\frac{p \rightarrow L_1^*,\ A,\ R_1^* \qquad q \rightarrow L_2^*,\ A,\ R_2^*}{p \rightarrow L_1^*,\ r,\ R_1^* \qquad r \rightarrow A \qquad q \rightarrow L_2^*,\ r,\ R_2^*}$$

In these rules lower-case letters denote non-terminal grammar symbols, upper-case letters – a sequence of one or more symbols (terminals or non-terminals) and upper-case

letters with an asterisk – a sequence of zero or more symbols (terminal or non-terminals). Both Intra-constuction and Inter-construction rules introduce new non-terminal symbols, while Absorption and Identification do not.

4. The Learning Process. The learning process consists of two major steps:

1. Initialization.
2. Application of the inference rules.

During step 1 an initial grammar rule set is formed. First we generate a set of new grammar rules – one for each example phrase in the training set. The left-hand side of each new rule is the start symbol and the right-hand side consists of the terminals (the words) in the example phrase. Thus, if we consider the example given in Section 2, the we obtain the following new rules:

(11) $s \rightarrow [a], [boy], [sings]$.
(12) $s \rightarrow [the], [boy], [sings], [a], [song]$.
(13) $s \rightarrow [the], [boy], [likes], [soccer]$.

Then we apply repeatedly the Absorption inference rule on the set of the new rules augmented with the dictionary rules until the new grammar rules are transformed into rules containing no terminal symbols in their right-hand side. In our example we obtain the following rules:

(14) $s \rightarrow det, n, v$.
(15) $s \rightarrow det, n, v, det, n$.
(16) $s \rightarrow det, n, v, n$.

The initial grammar rule set consists of the newly generated rules and the background rules. In our example these are rules 1–10 and 14–16.

The second step of the learning is in repeatedly applying any of the inference rules. In this process if we obtain identical rules we remove the duplicating ones (this also applies to step 1). Thus, if we apply Absorption on rules 9–10 and 14–16 and remove the duplicates, we obtain the following rules:

(17) $s \rightarrow np, v$.
(18) $s \rightarrow np, v, np$.

Then if we use Intra-construction on rules 17–18, the result is:

(19) $s \rightarrow np, vp$.
(20) $vp \rightarrow v$.
(21) $vp \rightarrow v, np$.

These rules 19–21 together with rules 1–10 form the final grammar generated in the learning process.

5. Conclusion. In this paper we present a method of learning CFG using the inference rules adapted from Inductive Logic Programming. Further research will be in the following directions:

- the use of both positive and negative training examples;
- defining a strategy for selecting the *best* inference rule to be applied in step 2 of the learning process;
- generalizing the problem to learn Definite Clause Grammar instead of pure context-free grammar;
- implementation of efficient algorithms in Prolog.

REFERENCES

[1] R. J. Mooney. Inductive logic programming for natural language processing. In: Proc. of the Sixth International Workshop on Inductive Logic Programming (Ed. S. Muggleton), Springer-Verlag, Berlin, 1997, 3–21.
[2] R. J. Mooney, M. E. Calif. Induction of first-order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, **3** (1995), 11–24.
[3] S. Muggleton, L. De Raedt. Inductive logic programming: theory and methods. *Journal of Logic Programming* **19, 20** (1994), 629–679.
[4] S. Muggleton. Inductive Logic Programming: issues, results and the challenge of Learning Language in Logic. *Artificial Intelligence Journal* (to appear).
[5] C. A. Thompson, R. J. Mooney, L. R. Tang. Learning to parse natural language database queries into logical form. Workshop on Automata Induction, Grammatical Inference and Language Acquisition, 1997.

Alexander Grigorov
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: grigorov@math.bas.bg

МАШИННО ОБУЧЕНИЕ НА КОНТЕКСТНО-СВОБОДНИ ГРАМАТИКИ ИЗПОЛЗВАЙКИ МЕТОДИ НА ИНДУКТИВНОТО ЛОГИЧЕСКО ПРОГРАМИРАНЕ

Александър Тодоров Григоров

В тази статия представяме един метод за машинно обучение на контекстно-свободни граматики използвайки методи, адаптирани от индуктивното логическо програмиране. За тази цел сме модифицирали правилата за извод, използвани в метода на обратна резолюция и ги прилагаме за машинно обучение на граматики. Статията описва постановката на задачата, правилата за извод и процеса на обучение.