Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 119 (2017) 73–82

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Young Scientists Conference in HPC and Simulation, YSC 2017, 1-3 November 2017, Kotka, Finland

Neural network for synthesizing deterministic finite automata

Petr Grachev[a], Igor Lobanov[b], Ivan Smetannikov[a], Andrey Filchenkov[a]

[a]*Computer Technology Lab, ITMO University, Kronverksky pr. 49, St. Petersburg, 197101, Russia*

[b]*Mathematical Research Methods of Complex Physical Systems Lab, ITMO University, Kronverksky pr. 49, St. Petersburg, 197101, Russia*

Abstract

Deterministic finite automata are widely used in control systems: from abstract protocols such as TCP to mechanical devices such as elevators or traffic lights. Some of these systems are quite complex and can be defined only in terms of formal language theory. In this paper, we propose new approach for synthesizing finite automata from a dictionary of some language that uses neural networks. The results show that the proposed approach works correctly and quickly for automata with up to six states and four characters in the alphabet. For larger automata, the neural network suffers from the vanishing gradient problem, which is a big topic for further research.

Peer-review under responsibility of the scientific committee of the 6th International Young Scientist conference in HPC and Simulation

Keywords: deterministic finite automaton; neural network; formal language; language dictionary

1. Introduction

Construction of a deterministic finite automaton according to a given formal language is a standard problem of formal language theory [1]. This problem is encountered in tasks related to formal languages, for example, to gain regular expressions. Methods of automata construction with respect to a formal language to a formal language are well-described in the literature [2, 3].

A plenty of methods for inferring an automaton exist, including SAT-solvers [4, 5, 6], optimization methods and meta-heuristics [7, 8, 9] and various other approaches [10, 11, 12].

This paper is devoted to construction of deterministic finite automaton according to a set of words belonging to some formal language using machine learning, namely, neural networks. In itself, the application of machine learning elements in problems associated with the formal languages theory is not new. The authors of [13] explore the task of translating natural language queries into regular expressions using neural generation. The authors of [14] de-

* Petr Grachev
E-mail address: grachev239@gmail.com

1877-0509
Peer-review under responsibility of the scientific committee of the 6th International Young Scientist conference in HPC and Simulation
10.1016/j.procs.2017.11.162

scribe method of using deep learning in the real task of constructing regular expressions. The idea of applying neural networks in also not new and is highlighted in the next Section.

The method we propose in this paper is capable to infer automata using relatively simple neural network architecture in an interpretable manner with clear ties to probabilistic relationship between automaton elements.

The rest of the paper is organized as follows. Section 2 contains brief description of automata and neural network concepts, as well as works that uses neural networks to synthesize automata. In Section 3, we describe the proposed neural network architecture, while Section 4 is devoted to the description of a learning procedure for this neural network. Section 5 contains experiment description, results of which are presented in Section 6. Section 7 concludes the paper with discussion and outline of the future work.

2. Background

2.1. Deterministic finite automata

Formal language is a set of finite words that consist of characters from some predetermined set called *alphabet* [15]. If an alphabet consists of two characters, which we will refer to as *'a'* and *'b'*, then it is possible to define a formal language, which contains all possible words that include these two characters and only them. It is not possible to define this language by naive enumeration of its words, because the number of the words in this language is infinite. However, one can simply verify that words *'a'*, *'aababab'* and *'bbbbb'* are included in this language and words *'c'*, *'abababa$'* and *'abAB'* are not.

Some formal languages that are called *regular* can be specified by a deterministic finite automaton (DFA) [16]. DFA is an abstract mathematical model, a structure consisting of a set of states and transitions between them. Transitions in DFA are labeled by characters of the alphabet. Each state is either *terminal* or *non-terminal*. If some word **w** was given as an input of some DFA and this automaton stopped in a terminal state after processing, then word **w** is considered to be included in regular language, which is determined by this DFA. On the contrary, if the DFA stopped in a non-terminal state, then word **w** is considered to be not included in corresponding language.

More formally, DFA is uniquely determined [17] by the five parameters: a finite alphabet Σ, a set of inner states Q, an initial state $s \in Q$, a set of terminal states $T \subset Q$, and a state transition function $\sigma : (Q \times \Sigma) \to Q$.

2.2. Neural Network

An *artificial neural network* is a mathematical model based on interconnected group of nodes called *neurons*. NN can be interpreted as a directed graph containing nodes and internodal connections. Signals are transmitted between pairs of nodes; sets of nodal outputs are created on the basis of inputs from other nodes [18].

In the *multilayer perceptron model* of NN, layers of neurons are placed between the input layer (containing input nodes) and the output neuron [19]. These neurons are connected by weights and output signals, which are a function of a sum of the inputs to the node modified by some *activation function* [20]. *Learning* of such a network can be viewed as a process of updating these weights in response to external stimuli so that it can perform a specific task [19]. *Feedforward* NN is a NN without closed paths in its topology [21]. If a network contains connections between the nodes that form a cycle then such a network is called *recurrent* NN. It is this type of network that underlies the proposed model.

2.3. NNs for synthesizing DFA

One of the variants of applying recurrent neural network for synthesizing of a finite-state automaton is applying the *first-order network* [22]. With this approach, the whole word is given as the input the neural network.

Another way of using RNN for DFA synthesis is implemented in *second-order networks*. In this case, the input word given as the input to network one character after another. The network model consists of an input neuron, one threshold unit, $|Q|$ state units and one output neuron. The output neuron and each state neuron receive a first order connection from the input neuron and the threshold neuron. In addition, each of the output and state neurons receives a second-order connection for each pairing of the input and threshold [23].