Forecast Method for Natural Language Constructions Based on a Modified Gated Recursive Block

Eugene Fedorov[1][0000-0003-3841-7373], Olga Nechyporenko[1][0000-0002-3954-3796], Tetyana Utkina[1][0000-0002-6614-4133]

[1] Cherkasy State Technological University, Cherkasy, Shevchenko blvd., 460, 18006, Ukraine

{fedorovee75, olne}@ukr.net,
t.utkina@chdtu.edu.ua

Abstract. The paper proposes a method for predicting natural language constructions based on a modified gated recursive block. For this, an artificial neural network model was created, a criteria for evaluating the efficiency of the proposed model was selected, two methods for parametric identification of the artificial neural network model were developed is based on the backpropagation through time algorithm and based on simulated annealing particle swarm optimization algorithms. The proposed model and methods for its parametric identification make it possible to more accurately control the share of information coming from the input layer and the hidden layer of the model, increase the parametric identification speed and the prediction probability. The proposed method for predicting natural language constructions can be used in various intelligent natural language processing systems.

Keywords: modified gated recursive block, prediction of natural language constructions, particle swarm optimization, simulated annealing, parametric identification.

1 Introduction

Currently, one of the most important problems in the field of natural language processing is the insufficiently high accuracy of the analysis of alphabetic and/or phoneme sequences [1, 2]. This leads to the fact that natural language processing may be ineffective. Therefore, the development of methods for predicting natural language constructions is an important task.

As a prediction method, a neural network forecast [3] was chosen, which, when forecasting natural language constructions, has the following advantages:

- correlations between factors are studied on existing models;
- no assumptions regarding the distribution of factors are required ;
- prior information about factors may be absent;
- source data can be highly correlated, incomplete or noisy;
- analysis of systems with a high degree of nonlinearity is possible;

- fast model development;
- high adaptability;
- analysis of systems with a large number of factors is possible;
- full enumeration of all possible models is not required;
- analysis of systems with heterogeneous factors is possible.

The following recurrent networks are most often used as forecast neural networks [4-6]:

- Jordon neural network (JNN) [7, 8];
- Elman neural network (ENN) or simple recurrent network (SRN) [9, 10];
- bidirectional recurrent neural network (BRNN) [11, 12];
- long short-term memory(LSTM) [13, 14];
- gated recurrent block (GRU) [15, 16];
- echo state network (ESN) [17, 18];
- liquid state machine (LSM) [19, 20].

Table 1 shows the comparative characteristics of neural networks for predicting natural language constructions.

Table 1. Comparative characteristics of neural networks for the predicting natural language constructions

Network **Criterion**	**JNN**	**ENN (SRN)**	**BRNN**	**LSTM**	**GRU**	**ESN**	**LSM**
Low probability of getting into a local extremum	-	-	-	-	-	+	+
High learning speed	+	+	+	-	+	+	+
Possibility of batch training	-	-	-	-	-	-	+
Dynamic control of the share of information from the input and hidden layers	-	-	-	+	+	-	-

According to table 1, none of the networks meets all the criteria. In this regard, the creation of training methods that will eliminate these drawbacks is relevant.

To increase the probability of falling into a global extremum and replacing batch training with multi-agent training, metaheuristic search is often used instead of local search [21-25]. Metaheuristics expands the capabilities of heuristics by combining heuristic methods based on a high-level strategy [26-30].

Modern metaheuristics may have one or more of the following disadvantages:

- there is only a generalized method structure or the method structure is focused on solving only a specific problem [21];
- iteration numbers are not present when searching for a solution [22];
- the method may not converge [31];
- material potential solutions are unacceptable [32];
- there is no formalized parameter values search strategy [33];
- the method is not intended for conditional optimization [34];

- the method does not possess high accuracy [35].

Thereby, arises the problem of constructing an effective metaheuristic optimization method.

Thus, the task to create an effective forecast model for alphabetic and/or phoneme sequences, which is trained based on effective metaheuristics, is relevant today.

The purpose of the work is to develop a forecast method for natural language constructions based on a modified gated recurrent block. To achieve the goal, the following tasks were set and solved:

1. Create a model of a modified gated recursive block.
2. Select a criteria for evaluating the efficiency of the proposed model.
3. Develop a method for the parametric identification of a model based on local search.
4. Develop a method for parametric identification of a model based on a multi-agent metaheuristic search.
5. Conduct a numerical study.

2 Creating a model of a modified gated recursive block

The paper proposes a modification of GRU by introducing $\left(1-r_j\left(n\right)\right)$ factor for the weighted sum of the neurons outputs in the input layer, which allows to more accurately control the share of information coming from the input layer and the hidden layer.

The proposed modified gated recurrent unit (MGRU) is a recurrent two-layered artificial neural network (ANN) with an input layer in, a hidden layer h, an output layer out. Just as for a regular GRU, each neuron in hidden layer is associated with reset and update gates is (FIR filters). The structural representation of the MGRU model is shown in Fig. 1.

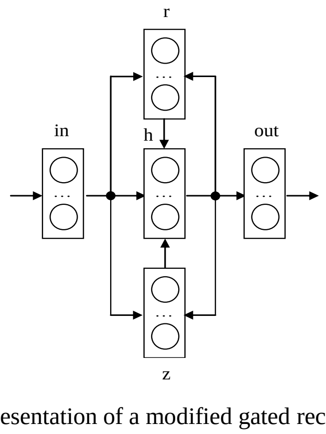

Fig. 1. Structural representation of a modified gated recurrent block (MGRU)

Gates determine how much information to pass. Thereby, the following special cases are possible. If the share of information passed by the reset gate is close to 0.5, and the share of information passed by the update gate is close to 0, then we get SRN. If the share of information passed by the reset gate is close to 0 and the share of information passed by the update gate is close to 0, then the ANN information is updated only due to the input (short-term) information. If the share of information passed by the update gate is close to 1, then the ANN information is not updated. If the share of information passed by the reset gate is close to 0 and the share of information passed by the update gate is close to 1, then the ANN information is updated only due to internal (long-term) information.

The modified gated recurrent block (MGRU) model is presented in the following form:

- calculating the share of information passed by the reset gate

$$r_j\left(n\right)=f\left(\sum_{i=1}^{N^{(0)}} v_{ij}^{in-r} y_i^{in}\left(n\right)+\sum_{i=1}^{N^{(1)}} v_{ij}^{h-r} y_i^{h}\left(n-1\right)\right),\ j\in\overline{1,\ N^{(1)}}\ ,$$

- calculating the share of information passed by the update gate

$$z_j\left(n\right)=f\left(\sum_{i=1}^{N^{(0)}} u_{ij}^{in-z} y_i^{in}\left(n\right)+\sum_{i=1}^{N^{(1)}} u_{ij}^{h-z} y_i^{h}\left(n-1\right)\right),\ j\in\overline{1,\ N^{(1)}}\ ,$$

- calculating the output signal of the candidate layer

$$\tilde{y}_j^h(n)=g\left((1-r_j(n))\sum_{i=1}^{N^{(0)}} w_{ij}^{in-h} y^{in}(n-i)+r_j(n)\sum_{i=1}^{N^{(1)}} w_{ij}^{h-h} y_i^h(n-1)\right),\ j\in\overline{1,\ N^{(1)}}\ ,$$

- calculating the output signal of the hidden layer

$$y_j^h(n)=z_j(n)y_j^h(n-1)+(1-z_j(n))\tilde{y}_j^h(n),\ j\in\overline{1,\ N^{(1)}}\ ,$$

- calculating the output signal of the output layer

$$y_j^{out}\left(n\right)=f\left(\sum_{i=1}^{N^{(1)}} w_{ij}^{h-out} y_i^{h}\left(n\right)\right),\ j\in\overline{1,\ N^{(2)}}\ ,$$

$$f\left(s\right)=\frac{1}{1+e^{-s}}\ ,$$

$$g\left(s\right)=\tanh\left(s\right),$$

where $N^{(0)}$ is the number of neurons in the input layer;

$N^{(2)}$ is the number of neurons in the output layer;

$N^{(1)}$ is the number of neurons in the hidden layer;

v_{ij}^{in-r}, u_{ij}^{in-z} are connection weight from the i th input neuron to the reset gates and update of the j th hidden neuron;

v_{ij}^{h-r}, u_{ij}^{h-z} are connection weight from the i th hidden neuron to the reset gates and update of the j th hidden neuron;

w_{ij}^{in-h} is connection weight from the i th input neuron to the j th hidden neuron;

w_{ij}^{h-h} is connection weight from the i th hidden neuron to the j th hidden neuron;

w_{ij}^{h-out} is connection weight from the i th hidden neuron to the output neuron;

$r_j(n)$ is share of information passed by the reset gate of the j th hidden neuron at a time n, $r_j(n) \in (0,1)$;

$z_j(n)$ is share of information passed by the update gate j th of the hidden neuron at a time n, $z_j(n) \in (0,1)$;

$y_i^{in}(n)$ is output of the i th input neuron at time n;

$y_i^{out}(n)$ is output of the i th output neuron at time n;

$y_j^{h}(n)$ is output of the j th hidden neuron at time n;

$f(\cdot)$, $g(\cdot)$ are activation function.

To evaluate the effectiveness of the proposed model, it is necessary to select criterion.

3 Selection of criterion for evaluating the effectiveness of the proposed model

In the paper, to evaluate the parametric identification of the MGRU model, a model adequacy criterion is chosen, which means the choice of such parameter values as $W = \left\{ v_{ij}^{in-r}(n), u_{ij}^{in-z}(n), w_{ij}^{in-h}(n), v_{ij}^{h-r}(n), u_{ij}^{h-z}(n), w_{ij}^{h-h}(n), w_{ij}^{h-out}(n) \right\}$, which deliver a minimum of the mean squared error (the difference between the model output and the desired output):

$$F = \frac{1}{PN^{(2)}} \sum_{p=1}^{P} \sum_{i=1}^{N^{(2)}} \left(y_{pi}^{out} - d_{pi} \right)^2 \to \min_{W} .$$

In the paper, to evaluate the functioning of the MGRU model in test mode, a forecast probability criterion is selected, which means the choice of such parameter values as $W = \left\{ v_{ij}^{in-r}(n), u_{ij}^{in-z}(n), w_{ij}^{in-h}(n), v_{ij}^{h-r}(n), u_{ij}^{h-z}(n), w_{ij}^{h-h}(n), w_{ij}^{h-out}(n) \right\}$, which deliver the maximum probability:

$$F=\frac{1}{P}\sum_{p=1}^{P}\rho\left(round\left(y_p^{out}\right),d_p\right)\to\min_W,\ \rho(a,b)=\begin{cases}1, & a=b\\ 0, & a\neq b\end{cases},$$

where $round(\cdot)$ is function that rounds a number to the nearest integer.

According to the first criterion, the methods of parametric identification of the MGRU model are proposed in this paper.

4 Creation of method for parametric identification of the MGRU model based on local search

In this paper, we first propose a method for parametric identification of the MGRU model based on the traditional for GRU backpropagation through time (BPTT).

The proposed method allows you to find a quasi-optimal vector of parameters' values of the MGRU model and consists of the following blocks.

Block 1 – Initialization:

- set the current iteration number n to one;
- initialization by uniform distribution over the interval $(0,1)$ or $[-0.5, 0.5]$ weights $v_{ij}^{in-r}(n)$, $u_{ij}^{in-z}(n)$, $w_{ij}^{in-h}(n)$, $i\in\overline{1,N^{(0)}}$, $j\in\overline{1,N^{(1)}}$, $v_{ij}^{h-r}(n)$, $u_{ij}^{h-z}(n)$, $w_{ij}^{h-h}(n)$, $i\in\overline{1,N^{(1)}}$, $j\in\overline{1,N^{(1)}}$, $w_{ij}^{h-out}(n)$, $i\in\overline{1,N^{(1)}}$, $i\in\overline{1,N^{(2)}}$, where $N^{(0)}$ is the number of neurons in the input layer, $N^{(2)}$ is the number of neurons in the output layer, $N^{(1)}$ is number of neurons in the hidden layer.

Block 2 – Setting the training set $\left\{\left(x_\mu,d_\mu\right)\middle|x_\mu\in R^{N^{(0)}},d_\mu\in R^{N^{(2)}}\right\}$, $\mu\in\overline{1,P}$, where x_μ is μ th training input vector, d_μ is μ th training output vector, P is the power of the training set. Number of the current pair from the training set $\mu=1$.

Block 3 is Initial calculation of the output signal for the hidden layer $h_i(n-1)=0$, $i\in\overline{1,N^{(1)}}$.

Block 4 is Calculation of the output signal for each layer (forward propagation)

$$y_i^{in}(n)=x_{\mu i},\ r_j(n)=f\left(s_j^r(n)\right),$$

$$s_j^r(n)=\sum_{i=1}^{N^{(0)}}v_{ij}^{in-r}(n)y_i^{in}(n)+\sum_{i=1}^{N^{(1)}}v_{ij}^{h-r}(n)y_i^h(n-1),\ j\in\overline{1,N^{(1)}},$$

$$z_j(n)=f\left(s_j^z(n)\right),$$

$$s_j^z(n)=\sum_{i=1}^{N^{(0)}}u_{ij}^{in-z}(n)y_i^{in}(n)+\sum_{i=1}^{N^{(1)}}u_{ij}^{h-z}(n)y_i^h(n-1),\ j\in\overline{1,N^{(1)}},$$

$$s_j^h(n)=\left((1-r_j(n))\sum_{i=1}^{N^{(0)}}w_{ij}^{in-h}(n)y_i^{in}(n)+r_j(n)\sum_{i=1}^{N^{(1)}}w_{ij}^{h-h}(n)y_i^h(n-1)\right),$$

$$j\in\overline{1,N^{(1)}},$$

$$y_j^h(n)=z_j(n)y_j^h(n-1)+\left(1-z_j(n)\right)g\left(s_j^h(n)\right),\ j\in\overline{1,N^{(1)}},$$

$$y_j^{out}(n)=f\left(s_j^{out}(n)\right),\ s^{out}(n)=\sum_{i=1}^{N^{(1)}}w_{ij}^{h-out}(n)y_i^h(n),\ j\in\overline{1,N^{(2)}},$$

$$f(s)=\frac{1}{1+e^{-s}},\ g(s)=\tanh(s).$$

Block 5 – Calculation of ANN error energy

$$E(n)=\frac{1}{2}\sum_{j=1}^{N^{(2)}}e_j^2(n),\ e_j(n)=y_j^{(L)}(n)-d_{\mu j}.$$

Block 6 – Setting up synaptic weights based on a generalized delta rule (back propagation)

$$w_{ij}^{h-out}(n+1)=w_{ij}^{h-out}(n)-\eta\,\Delta w_{ij}^{h-out}(n),\ i\in\overline{1,N^{(1)}},\ j\in\overline{1,N^{(2)}},$$

$$w_{ij}^{in-h}(n+1)=w_{ij}^{in-h}(n)-\eta\,\Delta w_{ij}^{in-h}(n),\ i\in\overline{1,N^{(0)}},\ j\in\overline{1,N^{(1)}},$$

$$w_{ij}^{h-h}(n+1)=w_{ij}^{h-h}(n)-\eta\,\Delta w_{ij}^{h-h}(n),\ i\in\overline{1,N^{(1)}},\ j\in\overline{1,N^{(1)}},$$

$$u_{ij}^{in-z}(n+1)=u_{ij}^{in-z}(n)-\eta\,\Delta u_{ij}^{in-z}(n),\ i\in\overline{1,N^{(0)}},\ j\in\overline{1,N^{(1)}},$$

$$u_{ij}^{h-z}(n+1)=u_{ij}^{h-z}(n)-\eta\,\Delta u_{ij}^{h-z}(n),\ i\in\overline{1,N^{(1)}},\ j\in\overline{1,N^{(1)}},$$

$$v_{ij}^{in-r}(n+1)=v_{ij}^{in-r}(n)-\eta\,\Delta v_{ij}^{in-r}(n),\ i\in\overline{1,N^{(0)}},\ j\in\overline{1,N^{(1)}},$$

$$v_{ij}^{h-h}(n+1)=v_{ij}^{h-h}(n)-\eta\,\Delta v_{ij}^{h-h}(n),\ i\in\overline{1,N^{(1)}},\ j\in\overline{1,N^{(1)}},$$

where η is a parameter that determines the learning speed (for large η, learning is faster, but the risk of getting the wrong decision increases), $0<\eta<1$.

$$\Delta w_{ij}^{h-out}\left(n\right)=y_i^h\left(n\right)\delta_j^{out}\left(n\right),$$

$$\Delta w_{ij}^{in-h}\left(n\right)=\left(1-r_j\left(n\right)\right)y_i^{in}\left(n\right)\delta_j^h\left(n\right),$$

$$\Delta w_{ij}^{h-h}\left(n\right)=r_j\left(n\right)y_i^h\left(n-1\right)\delta_j^h\left(n\right),$$

$$\Delta u_{ij}^{in-z}\left(n\right)=y_i^{in}\left(n\right)\delta_j^z\left(n\right),$$

$$\Delta u_{ij}^{h-z}\left(n\right)=y_i^h\left(n-1\right)\delta_j^z\left(n\right),$$

$$\Delta \nu_{ij}^{in-r}\left(n\right)=y_i^{in}\left(n\right)\delta_j^r\left(n\right),$$

$$\Delta \nu_{ij}^{h-r}\left(n\right)=y_i^h\left(n-1\right)\delta_j^r\left(n\right),$$

$$\delta_j^{out}\left(n\right)=f'\left(s_j^{out}\left(n\right)\right)\left(y_j^{out}\left(n\right)-d_{\mu j}\right),$$

$$\delta_j^h\left(n\right)=g'\left(s_j^h\left(n\right)\right)\left(1-z_j\left(n\right)\right)\left(\sum_{l=1}^{N^{(2)}}w_{jl}^{h-out}\left(n\right)\delta_l^{out}\left(n\right)\right),$$

$$\delta_j^z\left(n\right)=f'\left(s_j^z\left(n\right)\right)\left(y_j^h\left(n-1\right)-g\left(s_j^h\left(n\right)\right)\right)\left(\sum_{l=1}^{N^{(2)}}w_{jl}^{h-out}\left(n\right)\delta_l^{out}\left(n\right)\right),$$

$$\delta_j^r(n)=f'(s_j^r(n))\left(\sum_{i=1}^{N^{(1)}}w_{ij}^{h-h}(n)y_i^h(n-1)-\sum_{i=1}^{M}w_{ij}^{in-h}(n)y_i^{in}(n)\right)\delta_j^h(n).$$

Block 7 – Verification of the termination condition

If $n \bmod P>0$, then increase the number of the training pair μ by one, increase the iteration number n by one, and go to block 4.

If $n \bmod P=0$ and $\frac{1}{P}\sum_{s=1}^{P}E\left(n-P+s\right)>\varepsilon$, then increase the iteration number n by one and go to block 2.

If $n \bmod P=0$ and $\frac{1}{P}\sum_{s=1}^{P}E\left(n-P+s\right)<\varepsilon$, then end.

To increase the probability of falling into a global extremum and the possibility of parallel training of a neural network, the second method of parametric identification is further proposed.

5 Creation of a method for parametric identification of the MGRU model based on multi-agent metaheuristic search

In this paper, we propose a method for parametric identification of the MGRU model based on simulated annealing particle swarm optimization (SAPSO).

The SAPSO method allows you to find a quasi-optimal vector of parameter values for the MGRU model and consists of the following blocks.

Block 1 – Initialization:

- set the current iteration number n to one;
- set the maximum number of iterations N;
- set swarm size K;
- set the dimension of the particle position M (corresponds to the number of the MGRU model parameters);
- position initialization x_k (corresponds to the parameters vector of the MGRU model)

$$x_k = (x_{k1}, \ldots, x_{kM}),\ x_{ij} = \left(x_j^{\max} - x_j^{\min}\right)U(0,1) + x_j^{\min},\ k \in \overline{1, K},$$

where $x_j^{\min}$, $x_j^{\max}$ are minimum and maximum values, $U(0,1)$ is function that provides the calculation of a uniformly distributed random variable on a segment $[0,1]$;

- initialization of a personal (local) best position x_k^{best}

$$x_k^{best} = x_k,\ k \in \overline{1, K};$$

- speed initialization v_k

$$v_k = (v_{k1}, \ldots, v_{kM}),\ v_{ij} = 0,\ k \in \overline{1, K};$$

- create an initial swarm of particles

$$Q = \left\{\left(x_k, x_k^{best}, v_k\right)\right\};$$

- determine the particle from the current population with the best position (corresponds to the best parameters vector of the MGRU model in target function)

$$k^* = \arg\min_{k \in \overline{1, K}} F(x_k),\ x^* = x_{k^*}.$$

Block 2 – Modification of the velocity of each particle using simulated annealing

$$r1_k = (r1_{k1}, \ldots, r1_{kM}),\ r1_{kj} \in \left\{U(0,1), C(0,1), N(0,1)\right\},\ k \in \overline{1, K},\ j \in \overline{1, M},$$

$$r2_k = \left(r2_{k1}, \ldots, r2_{kM}\right),\ r2_{kj} \in \left\{U(0,1), C(0,1), N(0,1)\right\},\ k \in \overline{1,K},\ j \in \overline{1,M},$$

$$v_k = w(n)v_k + \alpha_1(n)\left(x_k^{best} - x_k\right)\left(r_1\right)^T + \alpha_2(n)\left(x^* - x_k\right)\left(r_2\right)^T,\ k \in \overline{1,K},$$

$$\alpha_1(n) = \alpha_2(n) = \alpha(0)\exp\left(-1/T(n)\right),\ \alpha(0) = \alpha_0 = 0.5 + \ln 2,$$

$$w(n) = w(0)\exp\left(-1/T(n)\right),\ w(0) = w_0 = \frac{1}{2\ln 2},$$

$$T(n) = \beta T(n-1),\ T(0) = T_0,\ \beta = N^{-\frac{1}{N-1}},\ T_0 = N^{\frac{N}{N-1}},$$

where $N(0,1)$ is function that provides the calculation of a random variable from standard normal distribution,

$C(0,1)$ is function that provides the calculation of a random variable from standard Cauchy distribution,

$\alpha_1(n)$ is parameter controlling the contribution of the component $\left(x_k^{best} - x_k\right)\left(r_1\right)^T$ to the particle’s velocity at the iteration n,

$\alpha_2(n)$ is parameter controlling the contribution of the component $\left(x^* - x_k\right)\left(r_2\right)^T$ to the particle’s velocity at the iteration n,

$w(n)$ is parameter controlling the contribution of the particle’s velocity at the iteration $n-1$ to the particle’s velocity at the iteration n,

α_0 is initial value of $\alpha_1(n)$ and $\alpha_2(n)$ parameters,

w_0 is initial value of $w(n)$ parameter,

$T(n)$ is annealing temperature at the iteration n,

T_0 is initial annealing temperature,

β is parameter that controls the annealing temperature.

The simulated annealing introduced in this work allow us to establish the inverse correlation between the parameters $\alpha_1(n)$, $\alpha_2(n)$, $w(n)$ and the iteration number, i.e. in the first iterations, the search is global, and in the last iterations, the search becomes local. In addition, in this work, a direct correlation between the parameters T_0, β and the iteration number is established, which allows for automated selection of these parameters.

The choice of initial values of $\alpha_0 = 0.5 + \ln 2$ and $w_0 = \frac{1}{2\ln 2}$ is standard and satisfies the conditions of a swarm convergence $w < 1$ and $w_0 > \frac{1}{2}\left(\alpha_1 + \alpha_2\right) - 1$.

Block 3 – Modification of each particle’s position, considering the limitations

$$x_k = x_k + v_k\,,\ k \in \overline{1, K}\,,$$

$$x_{kj} = \begin{cases} x_j^{\min}, & x_{kj} \le x_j^{\min} \\ x_{kj}, & x_{kj} \in \left(x_j^{\min}, x_j^{\max}\right) \\ x_j^{\max}, & x_{kj} \ge x_j^{\max} \end{cases},\ k \in \overline{1, K}\,,\ j \in \overline{1, M}\,,$$

$$v_{kj} = \begin{cases} v_{kj}, & v_{kj} \in \left(x_j^{\min}, x_j^{\max}\right) \\ 0, & v_{kj} \in \left\{x_j^{\min}, x_j^{\max}\right\} \end{cases},\ k \in \overline{1, K}\,,\ j \in \overline{1, M}\,.$$

Block 4 – Determining the personal (local) best position of each particle

If $F\left(x_k\right) \le F\left(x_k^{best}\right)$, then $x_k^{best} = x_k\,,\ k \in \overline{1, K}\,.$

Block 5 – Determining the particle from the current population with the best position

$$k^* = \arg \min_{k \in \overline{1, K}} F\left(x_k\right).$$

Block 6 – Determining the global best position

If $F\left(x_{k^*}\right) < F\left(x^*\right)$, then $x^* = x_{k^*}\,.$

Block 7 – Stop Condition

If $n < N$, then increase the iteration number n by 1 and go to block 2.

The proposed method is intended for implementation through a multi-agent system.

6 Numerical study

Table 2 presents the used lexemes and their categories, moreover the case when there is no lexeme is taken into account too.

Table 2. Lexemes and their categories

Lexeme category	Lexeme	Lexeme category	Lexeme
E	empty lexeme	V-INTR	think, sleep
N-H	man, woman	V-TR	see, chose
N-AN	cat, mouse	V-AGP	move, break
N-IN	book, rack	V-P	smell, see
N-AGR	dragon, monster	V-D	break, smash
N-FR	plate, glass	V-EAT	eat
N-F	break, cookie		

Before starting the parametric identification of the MGRU model, each letter of each lexeme is encoded. Table 3 presents the codes for each letter of the English alphabet, as well as a space.

Table 3. Letter codes

Letter	Code	Letter	Code	Letter	Code	Letter	Code
Space	00000	g	00111	n	01110	u	10101
a	00001	h	01000	o	01111	v	10110
b	00010	i	01001	p	10000	w	10111
c	00011	j	01010	q	10001	x	11000
d	00100	k	01011	r	10010	y	11001
e	00101	l	01100	s	10011	z	11010
f	00110	m	01101	t	10100		

In this work, a neural network forecast of the third lexeme by the previous two lexemes was carried out. Lexemes combination patterns are presented in table 4.

Table 4. Lexemes combination templates

First lexeme category	Second lexeme category	Third lexeme category
N-H	V-EAT	N-F
N-H	V-P	N-IN
N-H	V-D	N-FR
N-H	V-INTR	E
N-H	V-TR	N-H
N-H	V-AGP	N-IN
N-H	V-AGP	E
N-AN	V-EAT	N-F
N-AN	V-TR	N-AN
N-AN	V-AGP	N-IN
N-AN	V-AGP	E
N-IN	V-AGP	E
N-AGR	V-D	N-FR
N-AGR	V-EAT	N-H
N-AGR	V-EAT	N-AN
N-AGR	V-EAT	N-F

The number of neurons in the input layer is calculated as

$$N^{(0)} = 2 \cdot MaxLenLexem \cdot LenCode\ ,$$

where $LenCode$ is code length of one letter (according to table 3, $LenCode = 5$),

$MaxLenLexem$ is maximum lexeme length (according to table 2 $MaxLenLexem = 7$).

If the lexeme is shorter than $MaxLenLexem$, it is completed with spaces on the right. An empty lexeme consists of only spaces.

The number of neurons in the output layer is calculated as

$$N^{(2)} = MaxLenLexem \cdot LenCode\ .$$

The parametric identification of the MGRU model was carried out for 10,000 training implementations based on the proposed multi-agent metaheuristic search.

Table 5 presents the forecast probabilities obtained for 1000 test implementations based on the proposed MGRU model and artificial neural networks' traditional models.

Table 6 presents the number of parameters (link weights) for the proposed MGRU model and artificial neural networks' traditional models, which is directly proportional to the computational complexity of parametric identification.

Table 5. Forecast probability

Network / Criterion	JNN	ENN (SRN)	BRNN	complete LSTM	GRU	MGRU
Forecast probability	0.8	0.85	0.9	0.95	0.93	0.95

Table 6. Number of parameters

Criterion / Network	Number of parameters
JNN	$N^{(1)} \cdot \left(N^{(0)} + 2 \cdot N^{(2)}\right)$
ENN (SRN)	$N^{(1)} \cdot \left(N^{(0)} + N^{(1)} + N^{(2)}\right)$
BRNN	$2 \cdot N^{(1)} \cdot \left(N^{(0)} + N^{(1)} + N^{(2)}\right)$
complete LSTM	$N^{(0)} \cdot N^{(1)} + M \cdot N^{(1)} \cdot \left(N^{(0)} + N^{(1)} + N^{(2)} + N^{(1)} \cdot M\right)$
GRU	$3 \cdot N^{(1)} \cdot \left(N^{(0)} + N^{(1)} + N^{(2)}\right)$
MGRU	$3 \cdot N^{(1)} \cdot \left(N^{(0)} + N^{(1)} + N^{(2)}\right)$

For a complete LSTM, GRU, MGRU, the number of neurons in the hidden layer is calculated as $N^{(1)} = N^{(0)}$.

For JNN, ENN (SRN), BRNN, the number of neurons in the hidden layer is calculated as $N^{(1)} = 2 \cdot N^{(0)}$.

For LSTM, the number of memory cells is set as $M = 1$.

According to tables 5-6, MGRU and complete LSTM give the best forecast probability results, but MGRU, unlike LSTM, has fewer parameters, i.e. less computational complexity.

The increase in accuracy of MGRU prediction was made possible by introducing a multiplier $(1 - r_j(n))$ for the weighted sum of the neurons outputs in the input layer. This allows to more accurately control the share of information coming from the input layer and the hidden layer, as well as by using the metaheuristic determination of the MGRU models parameters.

The limitations of this work include the MGRUs full connectivity, the requirement for more parameters than in ENN (SRN), MGRU testing only on trigrams.

Like the BERT system, the proposed MGRU can work with context-free and context-sensitive grammars, but unlike BERT, it can be used not only for English.

The practical contribution of this work consists in the fact that it allows to predict alphabetic and / or phoneme sequences through an artificial neural network, the training of which is based on the proposed metaheuristics, which allows to increase the accuracy of the forecast and can be used as an intermediate stage in the speech understanding system.

7 Conclusions

1. To solve the problem of insufficient quality of the natural language sequences analysis, the corresponding neural network forecast methods were studied. To increase the efficiency of training neural networks, metaheuristic methods were studied.
2. The created model of the modified gated recurrent block allows for more precise control of the share of information coming from the input layer and the hidden layer, which increases the forecast accuracy.
3. The created method of parametric identification of the MGRU model based on simulated annealing particle swarm optimization reduces the probability of getting into local extremum and replaces batch training with multi-agent training, which increases the forecast probability and the training speed.
4. The proposed method for predicting natural language constructions based on a modified gated recurrent block can be used in various intelligent natural language processing systems.

References

1. Dominey, P.F., Hoen, M., Inui, T.: A Neurolinguistic Model of Grammatical Construction Processing. Journal of Cognitive Neuroscience. 18 (12), 2088–2107 (2006). doi: 10.1162/jocn.2006.18.12.2088
2. Khairova, N., Sharonova, N.: Modeling a Logical Network of Relations of Semantic Items in Superphrasal Unities. In: Proc. of the EWDTS. pp. 360–365. Sevastopol (2011). doi:10.1109/EWDTS.2011.6116585
3. Lyubchyk, L., Bodyansky, E., Rivtis, A.: Adaptive Harmonic Components Detection and Forecasting in Wave Non-Periodic Time Series using Neural Networks. In: Proc. of the ISCDMCI’2002. pp. 433–435. Evpatoria (2002).
4. Du, K.-L., Swamy, K.M.S.: Neural Networks and Statistical Learning. Springer-Verlag, London (2014). doi: 10.1007/978-1-4471-5571-3
5. Haykin, S.: Neural Networks. Pearson Education, NY (1999).
6. Sivanandam, S.N., Sumathi, S., Deepa, S.N.: Introduction to Neural Networks using Matlab 6.0. The McGraw-Hill Comp., Inc., New Delhi (2006).
7. Jordan, M.I.: Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In: Proc. of the Ninth Annual Conference of the Cognitive Science Society. pp. 531–546. Hillsdale, NJ (1986).
8. Jordan, M., Rumelhart, D.: Forward Models: Supervised Learning with a Distal. Cognitive Science. 16, 307–354 (1992). doi: 10.1016/0364-0213(92)90036-T

9. Zhang, Z., Tang, Z., Vairappan, C.: A Novel Learning Method for Elman Neural Network using Local Search. Neural Information Processing – Letters and Reviews. 11 (8), 181–188 (2007).
10. Wiles, J., Elman, J.: Learning to Count without a Counter: a Case Study of Dynamics and Activation Landscapes in Recurrent Networks. In: Proc. of the Seventeenth Annual Conference of the Cognitive Science Society. pp. 1200–1205. Cambridge, MA (1995).
11. Schuster, M., Paliwal, K.K.: Bidirectional Recurrent Neural Networks. IEEE Transactions on Signal Processing. 45 (11), 2673–2681 (1997). doi:10.1109/78.650093
12. Baldi, P., Brunak, S., Frasconi, P. Soda, G., Pollastri, G.: Exploiting the Past and the Future in Protein Secondary Structure Prediction. Bioinformatics. 15 (11), 937–946 (1999). doi: 10.1093/bioinformatics/15.11.937
13. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Technical Report FKI-207-95, Fakultat fur Informatik, Technische Universitat Munchen. doi: 10.1162/neco.1997.9.8.1735
14. Gers, F.: Long Short-Term Memory in Recurrent Neural Networks. PhD thesis, Ecole Polytechnique Federale de Lausanne.
15. Cho, K., Merrienboer, van B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Qatar, Doha (2014). doi: 10.3115/v1/D14-1179
16. Dey, R., Salem, F.M.: Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. (2017). arXiv: 1701.05923
17. Jaeger, H.: A Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the “Echo State Network” Approach. GMD Report 169, Fraunhofer Institute AIS (2002).
18. Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U.: Optimization and Applications of Echo State Networks with Leakyintegrator Neurons. Neural Networks. 20 (3), 335–352 (2007). doi: 10.1016/j.neunet.2007.04.016
19. Maass, W., Natschläger, T., Markram, H.: Real-Time Computing without Stable States: a New Framework for Neural Computation Based on Perturbations. Neural Computation. 14 (11), 2531–2560 (2002). doi: 10.1162/089976602760407955
20. Jaeger, H., Maass, W., Prıncipe, J.C.: Special Issue on Echo State Networks and Liquid State Machines. Editorial. Neural Networks. 20 (3), 287–289 (2007). doi: 10.4249/scholarpedia.2330
21. Talbi, El-G.: Metaheuristics: from Design to Implementation. Wiley & Sons, Hoboken, New Jersey (2009).
22. Engelbrecht, A.P.: Computational Intelligence: an Introduction. Wiley & Sons, Chichester, West Sussex (2007).
23. Zbigniew, J.C.: Three Parallel Algorithms for Simulated Annealing. In: Proceedings of the 4th International Conference on Parallel Processing and Applied Mathematics-Revised Papers (PPAM’01). pp. 210–217. Springer-Verlag, Berlin, Heidelberg (2001).
24. Loshchilov, I.: CMA-ES with Restarts for Solving CEC 2013 Benchmark Problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC’2013). pp. 369–376. Cancun, Mexico (2013).
25. Byrne, J., Hemberg, E., Brabazon, A., O’Neill, M.: A Local Search Interface for Interactive Evolutionary Architectural Design. In: Proceedings of the International Conference on Evolutionary and Biologically Inspired Music and Art (Evo-MUSART’2012). pp. 23–34. Springer, Berlin, Heidelberg (2012).

26. Yuen, S.Y., Chow C. K.: A Genetic Algorithm that Adaptively Mutates and Never Revisits. IEEE Transactions on Evolutionary Computation. 13 (2), 454-472 (2009). doi: 10.1109/TEVC.2008.2003008
27. Ventresca, M., Tizhoosh, H.R.: Simulated Annealing with Opposite Neighbors. In: Proceedings of the 2007 IEEE Symposium on Foundations of Computational Intelligence. pp. 186–192. Honolulu, HI, 2007. doi: 10.1109/FOCI.2007.372167
28. Wang, H., Li, H., Liu, Y., Li, Ch., Zeng, S.: Opposition-based Particle Swarm Algorithm with Cauchy Mutation. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation. pp. 4750-4756. Singapore (2007). doi: 10.1109/CEC.2007.4425095
29. Grygor, O.O., Fedorov, E.E., Utkina, T.Yu., Lukashenko, A.G., Rudakov, K.S., Harder, D.A., Lukashenko, V.M.: Optimization Method Based on the Synthesis of Clonal Selection and Annealing Simulation Algorithms. Radio Electronics, Computer Science, Control. 2, 90–99 (2019). doi: 10.15588/1607-3274-2019-2-10
30. Rasdi, M.R.H.M., Musirin, I., Hamid, Z.A., Haris H.C.M.: Gravitational Search Algorithm Application in Optimal Allocation and Sizing of Multi Distributed Generation. In Proceedings of the 2014 IEEE 8th International Power Engineering and Optimization Conference (PEOCO’2014). pp. 364–368. Langkawi, Malaysia (2014). doi: 10.1109/PEOCO.2014.6814455
31. Radosavljević, J., Jevtić, M., Klimenta, D.: Energy and Operation Management of a Microgrid using Particle Swarm Optimization. Engineering Optimization, 48 (5), 811–830 (2016). doi: 10.1080/0305215X.2015.1057135
32. Alinejad-Beromi, Y., Sedighizadeh, M., Sadighi, M.: A Particle Swarm Optimization for Sitting and Sizing of Distributed Generation in Distribution Network to Improve Voltage Profile and Reduce THD and Losses. In: Proceedings of the 2008 43rd International Universities Power Engineering Conference. pp. 1-5. Padova, Italy (2008). doi: 10.1109/UPEC.2008.4651544
33. Petrović, M., Petronijević, J., Mitić, M., Vuković, N., Miljković, Z., Babić, B.: The Ant Lion Optimization Algorithm for Integrated Process Planning and Scheduling. Applied Mechanics and Materials. 834, 187–192 (2016). doi: 10.4028/www.scientific.net/amm.834.187
34. Gandomi, A.H., Alavi, A.H.: Krill Herd: a New Bio-inspired Optimization Algorithm. Communications in Nonlinear Science and Numerical Simulation. 17 (12), 4831–4845 (2012). doi: 10.1016/j.cnsns.2012.05.010
35. Balochian, S., Ebrahimi, E.: Parameter Optimization via Cuckoo Optimization Algorithm of Fuzzy Controller for Liquid Level Control. Journal of Engineering. Hindawi Publishing Corporation (2013). doi: 10.1155/2013/982354