The Application of Neural Networks in the Construction of a Program for the Markup of Text

D. O. Zhaxybayev, darhan.03.92@mail.ru, West Kazakhstan Agrarian and Technical University named after Zhangir Khan, Uralsk, Kazakhstan

Corresponding author:

Zhaxybayev Darkhan O., Master of Pedagogical Sciences, Lecturer, West Kazakhstan Agrarian and Technical University named after Zhangir Khan, Uralsk, Kazakhstan
E-mail: darhan.03.92@mail.ru

Received on December 19, 2021
Accepted on January 19, 2022

This paper explores the use and applications of neural networks in the construction of a text markup program. This research paper describes various tasks that require textual data analysis and discusses the problems, issues, and solutions that accompany them. Interest in neural networks has increased in recent years, and they are finding applications in a wide variety of fields, such as business, medicine, engineering, geology, and physics. Neural networks have made great strides in forecasting, planning, and management. There are several reasons for this situation. Neural networks are very powerful modeling systems capable of creating complex dependencies. Neural networks can be widely used in areas such as text/speech recognition, semantic search, decision support/expert systems, inventory forecasting, data storage systems and content analysis. The object of the work is the process of functioning of neural networks and an algorithm for text markup recognition. The purpose of the scientific paper is the application of neural networks in the construction of the program for the markup of the text. In order to achieve the goal, the following tasks were put forward: a) study existing neural networks, b) choose a neural network to create a model and study its structure, c) convert input data to feed it into a neural network model. Representation and analysis of symbolic structures in neural networks seems to be an interesting and useful direction in neural network theory. In this paper, we have reviewed some neural network architectures that may merit further consideration in these circumstances. In what follows, we will focus on specific experiments in this area. Thus, this paper outlines a problem area for further research and testing.

Keywords: *neural networks, text markup, text data, neural network testing*

УДК 004.85 DOI: 10.17587/prin.13.142-147

Д. О. Жаксыбаев, магистр пед. наук, препод., darhan.03.92@mail.ru,
Западно-Казахстанский аграрно-технический университет имени Жангир хана,
Уральск, Казахстан

Применение нейронных сетей при построении программы для разметки текста

Исследуется использование нейронных сетей при построении программы для разметки текста. Описаны различные задачи, требующие анализа текстовых данных, обсуждены проблемы и решения, которые их сопровождают. В последние годы возрос интерес к нейронным сетям, они находят применение в самых разных областях, таких как бизнес, медицина, инженерия, геология и физика. С использованием нейронных сетей были достигнуты успехи в вопросах прогнозирования, планирования и управления. Причин такого эффекта несколько. Нейронные сети — очень мощные системы моделирования, способные создавать сложные зависимости. Нейронные сети могут широко использоваться в таких областях, как распознавание текстов / речи, семантический поиск, поддержка принятия решений / экспертные системы, прогнозирование запасов, системы хранения данных и анализ контента. Объектом работы является описание процесса функционирования нейронных сетей. Предметом научной работы является

алгоритм распознавания разметки текста. Целью научной статьи является изучение возможности применения нейронных сетей при построении программы для разметки текста. Для достижения описанной цели были поставлены следующие задачи: а) изучить существующие нейронные сети; б) выбрать нейронную сеть для создания модели и изучить ее строение; в) преобразовать входные данные для подачи их в модель нейронной сети. Представление и анализ символьных структур в нейронных сетях кажется интересным и полезным направлением в теории нейронных сетей. В статье проанализированы некоторые архитектуры нейронных сетей, которые, возможно, заслуживают дальнейшего рассмотрения. В будущем планируется сосредоточиться на конкретных экспериментах в этой области. Таким образом, данная работа очерчивает проблемную область для дальнейшего исследования и тестирования.

Ключевые слова: *нейронные сети, разметка текста, текстовые данные, тестирование нейронных сетей*

Introduction

In recent years there has been an increased interest in neural networks, which are successfully used in various fields — business, medicine, engineering, geology, physics. Neural networks have made great strides in solving prediction, planning, and control problems. There are several reasons for this shocking development.

1. Huge capabilities. Neural networks are a very powerful modeling system that can create complex dependencies.

2. Ease of use. Neural networks are learned through examples. The neural network user takes proxy data and then runs a learning algorithm that sees only the structure of the data. This, of course, requires the user to have heuristic knowledge of selecting and preparing the data, selecting the necessary network architecture, and interpreting the results, but must have the level of knowledge necessary to successfully use a neural network. For example, it is easier than using traditional numerical methods.

The neural network is intuitively interesting because it is based on the old biological mode of operation of the nervous system. In the future, the development of such neurobiological forms may lead to the creation of real "imaginary" computers.

Artificial neural network is based on the basic biological neural network, which is a neural network that performs certain functions. A neural network consists of neurons.

Synapses are connections through which signals from some neurons arrive at the input of others. A set of synapses is characterized by its weight. Those connected with a good weight are called excited, and those with an insufficient weight are called inhibited. An offshoot of neurons is called an axon. In an artificial neural network, the artificial neuron is a nonlinear function where the conflict is a combination of the lines of all signals. This function is called activation. The result of the activation function is then sent to the output neuron. By combining these neurons with others, it creates an artificial neural network.

Neurons perform several functions:

- reception function — synapses receive information;
- integration function — when a neuron is released, the signal carries information about all the signals combined in the neuron;
- transmission function— information is transferred from axon to synapse;
- translation function — a pulse reaching the end of the axon causes the transmitter to send a signal to the next neuron.

Applications of neural networks are diverse: message and speech recognition, semantic search, expert and decision support systems, stock price forecasting, data storage systems and content analysis.

The proliferation of computer and communication technologies has resulted in powerful systems and information flows unparalleled in the past. For example, the daily volume of records on electronic media does not allow even one person to access them immediately. The division of labor among several actors creates a problem of communication and organization between them.

At the same time, content analysis can be an important way to obtain relevant information that is important for a company to gain a competitive advantage. Many companies have established data protection departments that monitor and evaluate information about the company and its competitors. Similar methods can be used, for example, to evaluate new employees. By learning how a person behaves on Internet forums, blogs and social networks, you can get an idea of their character.

In this article we look at text markup, which is based on semantics. Semantic text markup as we understand it is a study by a computer of text that reveals the dependencies and combinations in certain words and expressions, within the text.

Obviously, it is not promising to constantly study the entire flow of information on a given topic and, incidentally, to form such a flow even without the use of automatic tools.

They can come in many forms, but we will consider the methods of these tools. Over the past two decades, methods of information production have evolved in the field of intelligent analysis and information extraction from data arrays. This is a field of study, created and developed on the scientific basis of exact sciences, such as statistics and artificial intelligence, theory of databases, etc. [1]. Originally, data mining was developed to produce highly organized data, but technological advances require the use of similar tools in unstructured data.

The presence of a large number of electronic documents stored in non-semantic tagged textual document formats indicates the importance of creating universal algorithms for automatic analysis of these documents to determine the semantics of individual document fragments.

The task is to select individual fragments from all text documents and assign these fragments to one of the predefined data types. The first information received is the

type of information for each fragment of its design, the presence of keywords or symbols, the position of other fragments in the corresponding text, etc. In general, there are no clear algorithms and rules for assigning an element to a fixed data type. The algorithm needed to solve such a problem allows decisions to be made under uncertain conditions. According to these algorithms, algorithms that perform mathematics using an artificial neural network have the ability to learn and make decisions based on the "knowledge" they acquire through hands-on learning [1].

Among the existing possibilities of using a neural network for color recognition, the use of a single-layer network consisting of an infinite number of neurons with a "backward distribution" learning algorithm is of great importance [2]. The theory of neural networks allows an infinite number of components and the number of neurons in each component, but in fact these numbers are limited by computer resources. A neural network consists of a set of neurons, each with a number of sharp spines, called spurs, and protruding axons. Each neuron is connected to synapses with input parameters of this neural network, each of which has its own weight. The process of training the neural network comes down to finding the ideal value for all synaptic embedded weights (some of which can be constant), which will provide the required response of the neural network. The ability of the network to perform operational tasks depends on how well it is trained. The quality of training depends on the duration of training and the order of replay during training, and is usually based on the appropriate ratio of training time to the acceptance of the final network performance. Thus, the ability of neural networks to make informed decisions depends more on the mapping of specific information to the network metrics on which the decision-making is based.

We consider the concept of text markup as syntactic and semantic processing of text in order to highlight and denote individual information objects.

Syntactic markup of text is one of the most complex, or morphosyntactic, markup means that in addition to the morphological information assigned to each word of the text, each sentence is also given its syntactic structure, and in the form of a dependency tree.

In order to use a neural network in text markup analysis, the following issues must be addressed:

- mapping the information coming into the neural network and ensuring that it is extracted from the file being examined;
- maintaining the structure of the neural network and the learning outcomes;
- taking steps to record the results as the neural network evolves.

Of course, there is a central module that provides an interface to all the other modules. Interaction between objects is done through a set of defined mechanisms — the programming interface. This feature makes it easy to upgrade the system by replacing a separate module for each system to update it or use it in a new environment. For example, a system may have several flexible user interface modules to work in different environments (Windows program, web interface, console program, web service), independent of other system modules. In addition, each of these modules must perform a number of specific tasks, for example: the choice of file for testing, determining the parameters of the algorithm, the report on the progress of reading algorithms, running training programs, etc. The well-known file integration module (I/O) uses a software interface to handle the file. It should provide a means to read information from the file and provide access to the file to open collection information to analyze and record results. This module can be added, for example, to Microsoft Word files and XML files. A set (or new ones) including the structure of the neural network, the description of the training result, and the collection and recording functions of the raw data are selected as parameters for the algorithm through the user interface. The test results are consistent and independent because the structure of the network depends on their number.

One example of randomly organized information is plain text. Another example (for semi-structured data [2]) is XML documents. Text is a global way of representing, collecting and transmitting information in society. Moreover, even this transformation is incapable of turning text into a relational representation without losing the semantics of the text and the relationship between entities. At the same time, a lot of information is hidden in the text, but due to the lack of structure it is impossible to evaluate it by the way it is obtained. This problem is solved by text analysis — Text Mining [2, 3].

Text Extraction — Text Mining

Let's take a look at the general practices of text mining [3]. These include general functions as well as very specific functions related to the type of data being studied. The first group includes planning and coordinating activities, the second group includes automatic annotation, recording key ideas, document navigation, trend analysis, and organization searches.

One of the most common tasks is organization (classification). The purpose of classification is to identify a set of components from a predetermined set to which a document belongs. Classification includes and other things that can be used for display.

The second task is content clustering. The purpose of this method is to automatically select identical groups of text between fixed sentences. Groups are formed based on only two identical text descriptions. No description of the group has been given.

Automatic annotation allows you to shorten notes and record their meaning. At the same time, the size of the sentences is usually adjustable by the user, allowing you to choose the amount of information you need.

If you take the main points, you can see the facts and context in the text. Often these ideas are proper nouns: names of people, names of organizations, etc.

Navigating between characters gives the user a fully coherent description of the type of text with clear themes and meaningful words. The user can select a subset of items of interest and switch between related links, for example by using automatic hyperlinks.

Trend analysis allows you to see certain patterns in text sentences. This can be used, for example, to measure how a company's interests change from one market segment to another.

When researching the analysis of organizational trends, it is necessary to examine the connections between key articles in the text.

Neural network methods

A number of strategies have been developed to solve the above problems. These tend to be the most common algorithmic methods. On the other hand, there are a number of powerful and effective solutions to some of these problems. These are neural network approaches. Neural network theory emerged and evolved as an attempt to explain the principles of the brain. Neural networks have proven to be a flexible and effective way to solve certain problems, which other methods cannot fully solve.

A neural network consists of organized neurons. Each neuron is a simple arithmetic object with many inputs and one output. Each input of the *i*-th neuron has a weight of synaptic signals w_i and x_i. Usually neurons compute weights by the weighted sum of inputs, and the results are influenced by another non-linear function *f* (neuron activation function) $y = f\left(\sum_i x_i w_i\right)$ (fig. 1).

Neurons are organized in fragments. Vector input $\mathbf{x} = \{x_i\}_0^N$ into each neuron in the cell (i. e. each neuron has N inputs). Release of all neurons at one stage serves as an input to the next stage. A neural network usually consists of several sections with feedback (repetition or perspective distribution), which exist as lessons (with or without a teacher) [5].

The use of neural networks to analyze datasets is not very common for several reasons. One of the main reasons is that the neural network is not set up to handle the data, but requires an access number. However, there are other options for this solution, and it could potentially play a key role in determining how our brains work with information. As you know, our thinking is symbolic (logical, we don't mean intuitive). There is a significant gap in our knowledge of how the brain works at a "lower" level (a good example of this mechanism is the neural network) and how our consciousness works. Perhaps someday this issue will be clarified by exploring the potential of the neural network to process text markup.

The most interesting neural network architectures suitable for processing text messages are the autonomous Kohonen map and the automatic associative memory network with repetition. Let's talk about them in more detail.

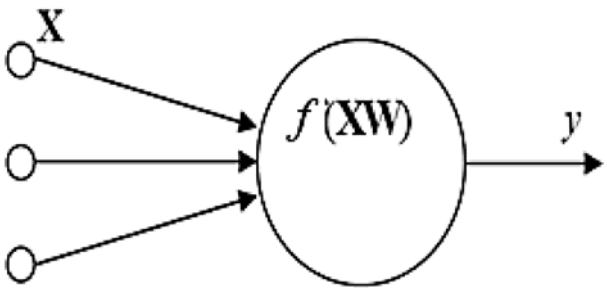

Fig. 1. Formal neuron with vector input X, vector weights W, activation function *f* and signal output *y*

Competitive Networks

The principles of operation and educational self-mapping were developed in 1982 by the Finnish scientist Teuvo Kalevi Kohonen. Kohonen's main idea is to provide information about his role in the regulation of learning neurons. According to Kohonen, a neural network consists of an input layer with the same number of neurons as the number of inputs, and a hidden (output) layer of one-dimensional (linear) or two-dimensional (rectangular) neurons. In relation to topographic maps such network is also called Kohonen map. The network architecture is shown on fig. 2 [6].

The $\|\,\mathrm{dist}\,\|$ box in this figure accepts the input vector $\mathbf{p}$ (R — number of elements in input vector) and the input weight matrix $\mathbf{IW}^{1,1}$, and produces a vector having S^1 elements. The elements are the negative of the distances between the input vector and vectors $_i\mathbf{IW}^{1,1}$ formed from the rows of the input weight matrix.

In this paradigm training is conducted without a teacher, which means that the neural performance was not comparable to the baseline at the time of the study. The training provides successive training examples to tune such a neural network. Another example identifies the most similar neuron, i.e. the neuron that is sent a smaller dot product of weight and vector as input. Such neuron is considered to be successful and adjusts the weight of neighboring neurons. The principle of learning, defined by Kohonen, provides competitive learning, taking into account the length of the "winning neuron" and neurons received as a result of recording as. $\Delta w_i^r = \beta\Lambda\left(|i - i^*|\right)\left(\mathbf{x}^r - \mathbf{w}_i\right)$, where $\Lambda(|i - i^*|)$ is the neighborhood function that determines the amount of neuron weight adjustment, $\mathbf{w}_i$ is the weight of the *i*-th neuron, and β is the learning rate.

At the end of the search process, examples of similarity clusters will be listed on the Kohonen map. Each set of neurons in the output layer was modeled to provide a form of training models in a multidimensional environment. The selected self-organizing map is included in the transformation of *N*-dimensional space into two-dimensional or one-dimensional space. The only thing to keep in mind is that such a change involves some errors.

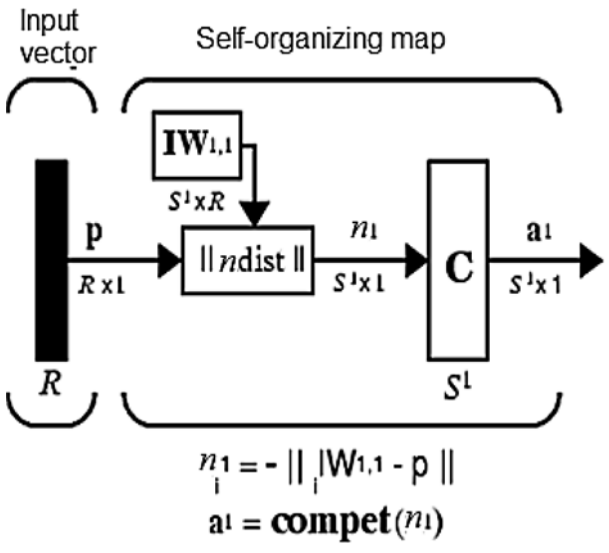

Fig. 2. Creating an independent Kohonen map

The two nearest points of the Kohonen map are closer to the *N*-dimensional input region, but not vice versa.

Based on this design, Finnish researchers have proposed solutions to the problems of coordination and movement of text in text boxes [7]. The way WEBSOMs are organized is that they present lines of full text as a two-sided map showing the distribution of text descriptions on the sides. In this case, specific documents are linked to their location on the map, and each location may contain a set of similar documents in a thematic text class. In addition, nearby areas are often associated with the closures of the text classes that make up the most important map. Places on the map are named according to contextual references. The user selects a location on the map of interest and gets a related text class with similar content. When searching for documents that contain other terms, search results can be displayed on a map that confirms the location of the documents. This allows the user to evaluate the thematic distribution of the required information.

An evolution of competitors, such as the Kohonen network, is the LVQ (Linear Vector Quantization) network. The LVQ neural network consists of two interconnected layers: the competitor construct and the conductive layer. Both areas of the LVQ neural network have a competitive neuron for each set and a linear neuron for each target class. S^1 is the number of clusters, S^2 is the number of target classes (S^1 is always greater than S^2). For example, suppose neurons 1, 2, and 3 in the competitive layer all learn subclasses of the input space that belongs to the linear layer target class 2. Then competitive neurons 1, 2, and 3 will have $\mathbf{LW}^{2,1}$ weights of 1.0 to neuron $\mathbf{n}^2$ in the linear layer, and weights of 0 to all other linear neurons. Thus, the linear neuron produces a 1 if any of the three competitive neurons (1, 2, or 3) wins the competition and outputs a 1. This is how the subclasses of the competitive layer are combined into target classes in the linear layer.

The competitive design divides the set of input vector sets **X** into classes and shows the regions between them for placing vectors $\mathbf{m}_i$. This is done by determining the distances between the *x*-value of the positions of the boundary segments and the initial value. The queue converts the input class (cluster $\mathbf{a}^1$) defined by the participant into the target user class $\mathbf{a}^2$ defined by the class. The output of all linear neurons is the corresponding $\mathbf{n}^2$, thus forming a binary vector $\mathbf{a}^2$, all elements of which are 0, except for the corresponding object class (this element is 1). Thus, the LVQ network consists of two parts: the competitive one, which

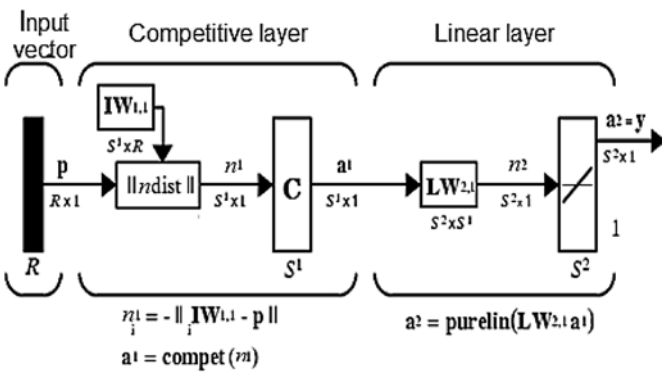

Fig. 3. LVQ network

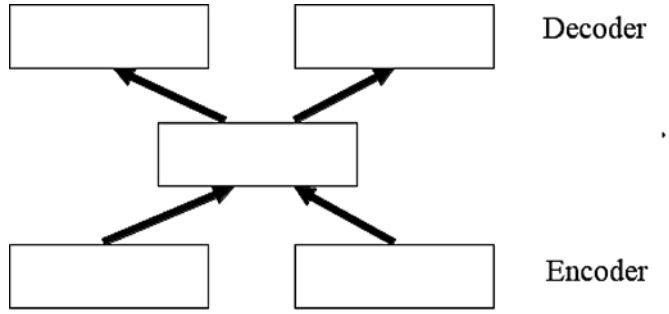

Fig. 4. RAAM binary network architecture

realizes the collection of access objects; the linear one, which connects groups of input objects with output classes. The architecture of the LVQ network is shown in fig. 3.

Recursive autoassociative memory

The third architecture we are interested in is recursive autoassociative memory (RAAM), developed by Pollack in 1990. The purpose of RAAM is to represent symbolic structures in a neural network. Symbols are stable forms of valid trees of fixed valence. In essence, a RAAM network is an automatic associative error back propagation network. The input and output fields of RAAM classify elements into fields, with each field containing the same number of elements. The number of fields is determined by the weight of the code trees, and the number of fields in the hidden area depends on the number of properties in each field. If we take a RAAM network that is trained to represent a set of poles, this network can be thought of as two automata: the first weight layer is the automaton for the production tree (called encoder) and the second layer is the dividing agent that represents the components (called decoder). A representative form of the RAAM architecture is shown in fig. 4 [7].

The representation of the structure in a RAAM network consists of the nature of the hidden event of this layer. When constructing a full tree view, RAAM creates a view for each inner hill (the bottom tree). Trees are repetitive forms, and RAAM creates its screen repeatedly, recreating the input design with a preconceived description of the components encountered at a given time.

On the Web, special indicators are represented as vectors. Orthogonal vectors are widely used. Measuring vectors representing signals shows the minimum number of elements that make up the RAAM field. Since the volume of the hidden component is smaller than the volume of the input material, the output signal is displayed in a compressed form. Thus, the degree of compression depends on the actual data set, and therefore it may be necessary to augment the vectors with zeros [8].

Hidden layer activation values form a distributed spatial representation of signals transmitted to the input network. The locations of different shapes and sizes can be determined using a self-organizing map to see the shape of their ends. It is still difficult to say what the result will be. However, we are convinced that such a function can be used, for example, to represent other structural shapes in texts.

Natasha Library for neurolinguistic programming

In our project we used a plug-in library called Natasha, written in the Python programming language. The goal of our project is to do syntactic and semantic markup of

the text. The first step in marking up a text is to find and identify all the verbs in the text. Then we mark the text semantically, i. e. we find the compatibility of verbs with adjacent words and classify these combinations.

Natasha is distributed under the MIT (Massachusetts Institute of Technology) license, i. e. open and free software license, which gives you the right to use it for your own purposes. The library itself is mostly written by two developers, natives of Russia. They have made all the source code and documentation for the library publicly available on GitHub. You can get access to the repository by following the link https://github.com/natasha/natasha.

Natasha solves the basic tasks of neurolinguistic programming (NLP) for the Russian language. These are at least the following tasks: tokenization, sentence segmentation, word embedding, morphology tagging, lemmatization, phrase normalization, syntactic analysis, Named Entity Recognition (NER) tagging, fact extraction. Natasha is not a research project; the underlying technologies are designed for use in real-world tasks. The developers have focused their attention on model size, memory usage and performance. NumPy is used to output information.

NumPy is a Python language library that adds support for large multidimensional arrays and matrices, along with a large library of high-level (and very fast) mathematical functions for operations on those arrays.

Natasha comes with an API which integrates quite a few written libraries. Using this API makes it easy to work with the module. Natasha includes the following libraries:

- Razdel — a system of lexemes of Russian sentences and words based on rules; it divides the text into words and sentences;
- navec — a set of compact pre-trained embeddings for the Russian language;
- SlovNet — modern deep learning methods for Russian NLP, compact models for Russian morphology, syntax;
- yargy — rule-based fact extraction;
- ipymarkup — neurolinguistic programming visualizations for named entity recognition and syntactic markup.

Using the Razdel library, we can break up text in sentences into individual words. It is also possible to divide text consisting of many sentences into a set of sentences (an array). For example, let's take the sentence "0.5L thermos mug (50/64 cm^3). Splitting the text into words, we get an array "['0.5', 'l', 'thermos', 'mug', '(', '50/64', 'cm^3',')']".

SlovNet is a Python library for NLP modeling based on deep learning for the Russian language. The library is integrated with other Natasha projects: Nerus, a large automatically annotated corpus, Razdel and Navec. Slovnet provides high-quality practical models for Russian NER, morphology and syntax. So, using SlovNet library we can perform morphological text parsing, breaking the whole text into words and get part of speech for each word. Also using this library we can perform syntactic parsing of a sentence [8].

Vargy uses rules and dictionaries to extract structured information from Russian text. I. e., using this library we can, for example, split the sentence "managing director Ivan Ulyanov" into an array "{ "person":{ "position": "managing director", "name":{ "first": "Ivan", "last": "Ulyanov" } } ".

Using the ipymarkup library we can find accessory words in a large volume of text. For example, we can find all the words that denote a person's identity, i. e. names and surnames. We can also find names of countries, cities, buildings, rivers, etc. [8].

In order to use Natasha in your application first you need to install the module in your system. And as we have already mentioned that the project is written in Python, so to install one must have Python at least version 3.5 and pip. First of all, download the Natasha project from GitHub and then run the command "pip install Natasha" in the root directory of the Natasha module. The installation of Natasha will also install other python libraries (dependencies) which are listed in the requirements directory.

To use the library in your application, as usual in the Python language, you need to import all the necessary Natasha library modules with the "import" statement [9].

Conclusion

Description of symbolic structures and their analysis with neural networks seems to be an interesting and useful guide in neural network theory. In our opinion, we have considered a number of neural network architectures that deserve further study in this context. Our future work focuses on specific experiments in this area. This defines the problem for further learning and practice, as well as the possibility of using other neural network modules to evaluate the displayed markup of text [6].

For citation:

Zhaxybayev D. O. The Application of Neural Networks in the Construction of a Program for the Markup of Text, *Programmnaya Ingeneria,* 2022, vol. 13, no. 3, pp. 142-147.

DOI: 10.17587/prin.13.142-147

References

1. **Chubukova I. A.** *Data Mining,* Moscow, BINOM, 2006, 382 p. (in Russian).
2. **Barsegyan A. A., Kupriyanov M. S., Stepanenko V. V., Holod I. I.** *Tehnologii analiza dannyh,* Saint Petersbirg, BHV-Peterburg, 2007, 384 p. (in Russian).
3. **Bashmakov A. I., Bashmakov I. A.** *Intellektual'nye informacionnye tehnologii,* Moscow, Izdatel'stvo MGTU im. N. E. Baumana, 2005, 304 p. (in Russian).
4. **WEBSOM —** Self-organizing Maps for Internet Exploration, available at: http://websom.hut.fi/
5. **Haykin S. O.** *Neural Networks and Learning Machines*, Pearson, 2009, 1104 p.
6. **Callan R.** *The Essence of Neural Networks*, Prentice Hall, 1998, 248 p.
7. **Barskij A. B.** *Logicheskie nejronnye seti:* Uchebnoe posobie, Moscow, Binom, 2013, 352 p. (in Russian).
8. **Galushkin A. I.** *Nejronnye seti: osnovy teorii,* Moscow, GLT, 2012. 496 p. (in Russian).
9. **Galushkin A. I., Cypkin Ja. Z.** *Nejronnye seti: istorija razvitija teorii:* Uchebnoe posobie dlja vuzov, Moscow, Al'jans, 2015, 840 p. (in Russian).