Development of a convolution algorithm for the content of a Web resource for processing by a neural network

S N Gorshkov\textsuperscript{1,2}, D V Solomonov\textsuperscript{1,3}, I V Dvoryaninova\textsuperscript{1,4} and R M Mihaylichenko\textsuperscript{1,5}

\textsuperscript{1}North-Caucasus Federal University, Stavropol, Russian Federation
\textsuperscript{2}E-mail: gorser96@gmail.com
\textsuperscript{3}E-mail: dsolomonov@ncfu.ru
\textsuperscript{4}E-mail: Innadv99@mail.ru
\textsuperscript{5}E-mail: furiousfaind@gmail.com

Abstract. The article aim to develop a convolution algorithm for the content of Web resources for processing by a neural network. The content of a web resource is represented in a vector form based on a co-occurrence matrix. A convolutional neural network, where the most significant features stand out from them, processes the content vectors of Web resources. The result is a collapsed content of Web resources in the form of a vector of significant features, which is the result.

1. Introduction
Web content is a set of textual and visual information. Such a set cannot be processed in the form in which it is presented to users, so it must first be converted into a form understandable to the computing device, and then be processed using a convolutional algorithm. Neural network processing requires that each word be converted into a form that is understandable to a computing device – a numeric one. For convenience of computations, it is necessary to represent one or several words in a vector form.

When processing content of Web resources, the problem of choosing the correct meaning of a word arises, therefore it is necessary to consider several words at once in order to more accurately determine the context of a fragment of a Web resource. One of the tasks of processing neural networks of texts is the construction of vectors of words related to the field of distributive semantics [4]. This task is based on the assumption that the meaning of a word can be understood by the meaning of its context, by the surrounding words.

2. Development of the algorithm for the vector transformation of the web resource content
Convolution of the content of Web resources is necessary to accelerate the processing of data by a neural network. The most effective means are convolutional neural networks. They convert a sequential set of vectors into a new fixed-length data set, without losing the meaning of the word layout.

When solving the task, first of all it is necessary to determine the method of converting text into a vector. The simplest representation of text in vector form is bag-of-words. In this case, on the basis of the learning corpus, a dictionary is constructed of all the words found in it or n-grams, where n is less than or equal to any predetermined value. The method in a primitive representation has several negative qualities.
When building a dictionary based on words, it is necessary to predict all possible words of Web content, which is impossible, and the volume of the dictionary will be too large. We will calculate the approximate data dictionary, if the dictionary consists of 100,000,000 words, then the vector will consist of 100,000,000 values, 1 vector element in the worst case is an integer value with a size of 4 bytes, which means the size of the vector will be 400,000,000 bytes = 381 MB. A large enough vector and its processing will greatly load the device. In addition to the problem of a potentially large vector size, there is a problem of loss of context and inefficient use of a vector, which will have only one number, and all other zeros.

This problem can be solved using n-grams and a co-occurrence matrix. If you use a 3-character n-gram and form a vector based on the co-occurrence matrix, then the generated dictionary will contain a maximum of 669920 elements, taking into account the special characters $C_{158}^3 = C_{3+158-1}^3 = \frac{(3+158-1)!}{3!(158-1)!} = 669920$.

Character n-grams are useful for languages with a rich morphology – the same words can be found in texts, but in different variations (different gender or numeric forms), but the root of the words does not change, and, consequently, the common set of characters.

Each n-gram must be saved to the database, but it is impossible to know in advance which n-grams will be found in the content of the Web resource, since the number and variety of the latter is too large. Therefore, a dictionary in the database will be formed as soon as a new n-gram is encountered in the next content of the Web resource.

Figure 1 shows the algorithm for the formation a database of key phrases.

![Algorithm for the formation of a database of key phrases](image)

**Figure 1.** Algorithm for the formation of a database of key phrases.

We describe the structure of the database of key phrases (figure 2). The database will contain 2 tables, the first table stores all known key phrases. The second table contains links between key phrases. The link between key phrases will be considered as a pair of adjacent key phrases in the content of a Web resource, a set of such links will reflect the context.

![Datalistic database model of key phrases](image)

**Figure 2.** Datalistic database model of key phrases.
The algorithm of vector formation is shown in figure 3. The algorithm transforms the array of key phrases into an array of corresponding vectors. As an array of key phrases, a sentence or another part of the content of a web resource, broken into symbolic 3 grams.

As a result of the operation of the algorithm, vectors with a length equal to the size of the dictionary of key phrases are obtained. The vector is filled according to the co-occurrence matrix. For example, for the phrase “neural net consists of neurons” a co-occurrence matrix will be formed.

**Table 1. Co-occurrence matrix.**

|   | neu | ral | net | co  | nsi | sts | of  | ne  | uro | ns  |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| neu| 0   | 2   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   |
| ral| 2   | 0   | 2   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| net| 0   | 2   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   |
| co | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| nsi| 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   |
| sts| 0   | 0   | 0   | 1   | 0   | 0   | 1   | 0   | 0   | 0   |
| of | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   |
| ne | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| uro| 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| ns | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Table 1 also presents the vectors for each symbolic 3-gram of the considered example, the vector is a sequence of numbers in the string. 3-gram character “net” will be converted into a vector: 

“net” = (0 2 0 1 0 0 0 0 0 0).
As can be seen from the example, most of the vector is zero, this part of the vector does not carry critical information about the context of the content of the Web resource, but will be involved in further processing, spending more productive resources of the target device. To solve this problem, it is necessary to isolate the main features of the content of the Web resource and form a smaller vector characterizing the content of the Web resource using a convolutional neural network.

3. Development of the algorithm of convolution and selection of significant features of the vector content of Web resources

Features of a convolutional neural network for word processing [9]:
- filter layer has the same width $m$ as the matrix;
- it is allowed to apply simultaneously different filter layers with different heights to highlight additional features.

A convolutional neural network is a very effective method for presenting texts. Due to the complexity of computing and storing data, most n-gram cases, including Google, do not represent n-gram cases with $n$ more than 5. For example, the Google n-gram case from 1,024,908,267,229 words, 95,119,665,584 sentences, 13,588,391 unigrams, 314,843,401 bigrams, 977,069,902 3-grams, 1,313,818,354 4-grams, 1,176,470,663 5-grams after compression in gzip format has a size of 24 GB [1].

A convolutional neural network for representing content of Web resources consists of two layers (figure 4): a convolutional layer and a pooling layer.

The convolution layer, in turn, consists of filters. A filter is a neuron, the input of which is formed with the help of windows moving through the text and choosing a certain number of words in succession [2]. At the output of the filter, one vector is formed, which aggregates all the word vectors in it. Then, one vector is formed on the pooling layer, corresponding to the entire sentence, which is calculated as an component-maximum from all the output filter vectors.

Consider the algorithm of the convolutional layer for a simple example (figure 5). In the convolutional layer of the neural network, vectors are processed using predefined filters [7]. The filter size is the number of simultaneously processed word vectors. Figure 5 shows the operation of the size 2 filter. In it, the scalar product with weights of the filter is calculated for each 2 vectors of input data. For example, calculate the first value of the output vector:

![Figure 4. The algorithm of the convolutional neural network for processing the vector content of Web resources.](image-url)
\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
\end{pmatrix}
\begin{pmatrix}
0.2 & 0.6 & 0.1 & 0.5 \\
-0.3 & 0.1 & 0.4 & 0.2 \\
\end{pmatrix}
= 
1 \cdot 0.2 + 0 \cdot 0.6 + 0 \cdot 0.1 + 0 \cdot 0.5 + 0 \cdot (-0.3) + 1 \cdot 0.1 + 1 \cdot 0.4 + 0 \cdot 0.2 = 0.7.
\]

**Figure 5.** Algorithm convolutional layer convolutional neural network.

Similarly, the remaining values of the new vector are calculated. The resulting vector is transferred to the next layer – the pooling layer. The pooling layer algorithm is depicted in figure 6.

**Figure 6.** The pooling layer algorithm.

The input pooling layer gets all the vectors of one filter from the previous layer and applies the component-wise maximum value function to them. Figure 6 shows the operation of the pooling layer for vectors (0.7 0.9 1.3 −0.1) and (1.2 0.3 1.1 0.4). After processing the layer obtained vector (1.2 0.9 1.3 0.4).

Training a convolutional neural network for the task is based on the method of training without a teacher. Hebb’s learning algorithm assumes that the synaptic connection of two neurons is enhanced if both of these neurons are excited.

The algorithm is expressed by equality (1), described in [3]:

\[
W_{ij}(t + 1) = W_{ij}(t) \ast (1 - \gamma) + \eta \ast IN(t) \ast OUT(t),
\]  

(1)
where $W_{ij}(t)$ is the synapse force from neuron i to neuron j at time $t$, $\gamma$ is the forgetting factor, is needed to stabilize the learning process, $\eta$ is the learning factor, the value of which is selected in the interval (0.1), IN(t) – input signal, OUT(t) – output signal. The value of the coefficient $\gamma$ is a part of the learning coefficient $\eta$.

The activation function is a function that calculates the output signal of an artificial neuron. The use of the activation function ReLU significantly increases the rate of convergence of the stochastic gradient descent compared to the sigmoid and hyperbolic tangent. A large gradient passing through the ReLU can lead to such an update of the balance that the neuron is never activated again. One of the modifications ReLU – Leaky ReLU, solves this problem. The activation function of Leaky ReLU is:

$$f(x) = \begin{cases} 
\alpha \times x, & x < 0 \\
0, & x \geq 0
\end{cases}$$

(2)

where $\alpha$ is a small constant.

The principle of choosing the initial values of weights for the layers of the neural network contributes to the speed of learning the network. The Ge initialization method makes it possible to simplify the passage of a signal through a layer, the activation function of which is the ReLU, which compensates for the fact that this function returns zero for half the definition domain. The initialization method of Ge has the form:

$$\text{Var}(w) = \frac{2}{n_{in}}$$

(3)

where $\text{Var}(w)$ – variance of random weight, $n_{in}$ – number of neurons in the previous layer.

4. Conclusion

As a result, an algorithm for checking the content of Web resources has been developed, which allows you to present any content in the form of vectors that modern computers can easily handle. After the content is verified, the most important features are preserved, initially being the links between the content fragments of the Web resource, which affects the context of the entire content in a minor way. This will allow in the future to effectively clustering the content of Web resources to detect illegal content in Web resources.

References

[1] Thorsten B and Franz A 2006 Web 1T 5-gram Version 1 LDC2006T13 DVD (Philadelphia: Linguistic Data Consortium)

[2] Yoav Goldberg 2017 Neural Network Methods for Natural Language Processing (Morgan & Claypool Publishers)

[3] Osovskii S 2002 Neural networks for information processing Translation from polish I D Rudinskii (Moscow: Finance and Statistics Publ)

[4] Information on https://www.osp.ru/os/2017/02/13052221

[5] Uossermen F 1992 Neurocomputer technology: Theory and practice (Moscow: Publ “Mir”)

[6] Krug P G 2002 Neural networks and neurocomputers: A manual for the course “Microprocessors” (Moscow: MEI Publ)

[7] Information on http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

[8] Chipiga A F and Kolkov D A 2006 Napravlenniy poisk globalnogo ekstremuma mnogomernih funktsiy Basic research

[9] Information on https://neurohive.io/ru/osnovy-data-science/7-arhitektur-nejronnyh-setej-nlp/