Methods of constructing a neural network for detecting abnormalities in data processing in the smart field application system

Marina Ganzhur^{1,}, Alexey Ganzhur¹, Ilya Borisenko¹, Andrey Kobylko¹, Dmitri Romanov¹*

¹Don State Technical University, 1, Gagarin sq., 344003, Rostov on Don, Russia

Abstract. The modern methods of analytics of user actions are considered; developed an algorithm for processing custom events for the SAP system; developed a neural network algorithm and analyzed its work results; options for improving the model are proposed.

1 Introduction

According to forecasts, closer to 2050, the humanity will need 70% more products for nutrition than today. But the deteriorating environmental situation, rise of energy costs and fall of soils fertility will become a serious obstacle to the production of the required amount of food. These problems can be solved by changing agricultural practices, in particular, by introducing the latest technologies and innovative solutions, such as the concept of the Internet of things.

Despite the fact that agriculture, due to the production specifics, seems a rather conservative industry to many people, it was one of the first ones where IT technologies were used, in particular, the Internet of Things (IoT). It bears reminding that IoT refers to the global concept of interaction and exchange of information between various devices, machines and systems via the Internet. It allows reducing human participation at some stages of production by automating the process and its control through various "smart" devices.

In the modern world, the business processes of any company are actively subjected to computer automation: electronic document management has been present on the market for a long time and new management technologies are increasingly introduced, one of which is SAP (Systeme, Anwendungen und Produkte - processing of data in systems, applications and products), which is the world leader in corporate applications in terms of revenue from software.

Almost every large enterprise has its own Security Operation Center (SOC) in order to:

- detect events, timely and in an automated fashion, that pose a potential threat to the organization, its information systems or information assets (information security incidents);
- ensure long-term storage of the entire volume of collected events and recorded information security incidents for the possibility of a post-incident investigation;

* Corresponding author: mganzhur@yandex.ru

- implement processing of identified incidents, which would allow, within the guaranteed time (depending on the level of the incident criticality), to notify the responsible departments of the organization about what happened, and recommend the required measures for preventing the impact of the information security incident on the business.

Relevance of the problem.

SAP is an extremely comprehensive automation system for production accounting at all stages of the life cycle of design and production. The automation of the processing, storage and transmission of information has led to the emergence of new problems related to its security. Today, information is one of the most valuable goods on the market and its role in society is not decreasing, and due to the current steady tendency of increasing the number of attacks on computer systems and networks, production automation systems are becoming the most interesting targets for malefactors. Mechanisms and methods of hacking are constantly being improved, and the existing means of protection simply do not keep pace with them.

1.1 Algorithm description

The main idea is to build a neural network, which, knowing the everyday behavior of the user and user's capabilities in the business space of the enterprise, will be able to detect unusual (abnormal) actions for this user in automatic or semi-automatic mode according to the above patterns of behavioral anomalies.

The object of analytics is the system user. The data source will be a log file that stores information about all the activities of the system users. Fortunately, SAP provides the ability to configure logging parameters for all types of logs: SAP Security Audit ABAP - Allgemeiner Berichts-Aufbereitungs-Prozessor, SAP Security Audit JAVA, SAP Security Audit HANA, SAP HTTP ABAP, SAP HTTP JAVA, SAP HTTP HANA, Gateway Log etc. The log allows the post-factum monitoring of the user's behavior to the smallest details: when the user has logged in; when the user has gone out; what reports/transactions were started, which tables were accessed; what settings were changed. All this makes it possible to compose the everyday context of the user's behavior and compare the user's current actions therewith. In the case of the author, the SAP Security Audit ABAP log will be considered.

It is assumed that ordered records from log files will be downloaded to the program. Each entry has a structure, as in Listing 1.

```
CEF:0|Comp SI|SAP Security Audit ABAP|1.5|ESC0012|Transaction Started|3|rt=apr
09 2017 11:29:17 msg=Transaction &A Started duser=SOME_USER src=211.163.214.153
cs2=002 cs2Label=sapclient001 destinationServiceName=GENERATION
dproc=SAPMSSY1 cat=AU3 action=SCCTY1 sapeventa=SCCTY1 cs1=DM0
cs1Label=sapsid dst=211.1.4.3 dhost=sapserver dvc=211.163.214.153
dvchost=GENERATION anomaly_status=None
```

Listing 1 - Structure of SAP Security Audit ABAP log.

Before loading this line to the neural network, it's important to identify the key fields for analytics:

1. cat - a field indicating the type of action and its execution result (successful/unsuccessful):

- Logon;
- Report Started;
- Transaction Started;
- RFC;
- etc.;

2. *sapevent [x]* (x - any lowercase letter of the Latin alphabet [a-z]) - a specific field that mainly contains information about the target, whether it is a program, transaction, report, table, or RFC;
3. *duser* - username at the SAP Application Server;
4. *rt* - time of event registration in the log;
5. *src* - IP-address of the user device;
6. *cs1* - individual entry point number (Instance) at the SAP Application Server.

The fields of *duser*, *src* and *cs1* help to separate users from each other in the system, but account names can coincide on different Instance, unlike their owners. The *cat*, *rt*, and *sapevent [x]* fields are used to create context and look for deviations in behavior. Therefore, the network will process the following sequences:

rt=apr 09 2017 11:29:17 duser= SOME_USER cat=AU3 sapeventa= SCCTY1

Listing 2 - Content of the LSTM vector.

1.2 Information coding

In order to load the information into the neural network, it should be digitized or encoded. Each of the fields which accept discrete values (*cat*, *sapevent [x]*) will be encoded in the float type and normalized (peak limiting effect): the encoded field values will take the form of a fractional number. The normalization involves the replacement of nominal characteristics so that each of them lies in the range from 0 to 1.

Some knowledge bases are preliminarily compiled containing all kinds of values for each of the field types (*cat*, *sapevent [x]*). Then, the data is sorted by the frequency of their mentioning in the log, designed for the neural network training.

Since time is a continuous quantity, it has to be encoded in the range from -1 to 1.

The normalization is needed, since most gradient methods (on which, in fact, almost all machine training algorithms are based) are highly sensitive to data scaling.

1.2.1 Encoding of time

From the *rt* field, only the time of the day is taken into account, i.e. hours (*h*), minutes (*m*) and seconds (*s*), with conversion to seconds:

$$rt' = h * 60 * 60 + m * 60 + s \quad (1)$$

Then *rt'* is normalized:

$$rt'' = \frac{rt'}{12*60*60} - 1. \quad (2)$$

After the normalization step, *rt''* will take values from -1 to 1.

1.2.2 Encoding the *cat* field

Each value will be assigned a number according to the dictionary sorted by the frequency of mentioning of all possible variations for a particular field in the training set (*cat.dict*) as follows:

$$cat'[cat] = \frac{cat.dict[cat]}{len(cat.dict)+1}, \quad (3)$$

where *cat.dict [cat]* is an integer equal to the serial number of *cat* in the *cat.dict* dictionary.

To focus more on sharp deviations in the user's behavior, rare and frequent events should be distributed in space. For this, nonlinear quadratic normalization is perfect:

$$cat'[cat] = \left(\frac{cat.dict[cat]}{len(cat.dict)+1} \right)^2. \quad (4)$$

Thus, the values of the *cat* fields will take values from 0 to 1.

1.2.3 Encoding the *sapevent[x]* field

By analogy with *cat*, each value of *sapevent[x]* will be assigned a number according to the *sapevent.dict* dictionary sorted by the frequency of mentioning of different variations for a certain field in the training set, as follows:

$$sapevent'[sapevent] = \frac{sapevent.dict[sapevent]}{len(sapevent.dict)+1}, \quad (5)$$

where *sapevent.dict [field]* is an integer equal to the sequence number of the *sapevent* value in the *sapevent.dict* dictionary.

To focus more on sharp deviations in the user's behavior, rare and frequent events should be distributed in space. For this, nonlinear quadratic normalization is perfect:

$$sapevent'[sapevent] = \left(\frac{sapevent.dict[sapevent]}{len(sapevent.dict)+1} \right)^2 \quad (6)$$

Thus, the values of the *sapevent [x]* fields will take values from 0 to 1.

1.3 Technical implementation of the work of the neural network algorithm

Each user of the system will involuntarily generate log entries by own actions. Accordingly, these records contain all the relationships and context that make up a pattern of behavior, which is very important for distinguishing normal actions for this user from abnormal ones.

To solve the problem of finding deviations, an algorithm is needed that can remember the context of the user's behavior. In predicting sequences, Deep Neural Networks have proven themselves well. The recurrent neural networks contain feedback and resemble a chain in the diagram below (Figure 1).

Fig. 1. Scheme of the operation of a recurrent neural network.

The recurrent neural networks potentially have the ability [23, 24] to connect the previous information with the current task, and they have already shown themselves well in many tasks in practice, such as speech recognition, language modeling, translation, image recognition and many others.

1.3.1 Problem of long-term dependencies

RNN (recurrent neural networks) are really capable of remembering sequences. However, when the data array and the pattern length are extremely large, the RNN may fail. As the distance between relevant information and the point of its application grows, conventional RNNs have the ability to forget information connections.

When training a neural network model, all the weights of the recurrent memory vector are overwritten at each iteration of the computational graph, therefore, problems with local dependencies may appear. This problem, namely the “problem of disappearing gradient”, was noticed and investigated in more detail by S. Hochreiter Y. Bengio, P. Frasconi, и J. Schmidhuber in some of their works: “Research for dynamic neural networks” [27] “Gradient flow in recurrent networks: the complexity of studying long-term dependencies” [28] and “On the complexity of recurrent neural networks training” [29].

Of course, at the correctly selected parameters, the RNN can still give us the desired result, but in practice, the selection of such parameters is extremely difficult. This problem was studied in detail by Sepp Hochreiter (1991) and Yoshua Bengio et al. (1994); in their work [19], they examined the problems of the operation of RNN with an increasing length of dependencies in detail.

1.3.2 LSTM network

The LSTM network was designed to address the aforementioned long-term dependencies problem.

The LSTM (long-short term memory) [12] is one of the many architectures of a recurrent neural network. It has special memory cells for storing information about past events. Its main advantage is the ability to detect true long-term data dependencies.

In LSTM networks, neurons are equipped with an integrated system of so-called gates and the concept of a cell state, which is a kind of long-term memory. The gates are some kind of control points that are responsible for working with information: which part in the cell state should be kept, which should be discarded and which should be passed to the output.

Any recurrent neural network is a chain of repeating neural network modules. In the simplest RNN, this module can have only one activation function (e.g. hyperbolic tangent) (see Figure 2), and in the case of LSTM, the modules are much more complex (see Figure 3). It has 4 layers of the neural network instead of one, in comparison with the ordinary RNN.

Fig. 2. Scheme of operation of a simple RNN with hyperbolic tangent as an activation function.

As it was previously described, the LSTM has gates (Figure 3).

Fig. 3. The LSTM network operation diagram.

In the figures above, the line symbolizes the vector transfer; the junction of two lines is equivalent to the concatenation of two vectors, and the branching - to copying; the rectangle is a neural network layer; the oval with operation inside is a corresponding operation with incoming vectors.

1.4 Detailed analysis of the LSTM operation algorithm

The whole essence of the LSTM network, like any other recurrent neural network, lies in the repeating modules of the RNN chain - the LSTM cell [13]. Figure 4 shows the detailed structure of the LSTM cell.

Fig. 4. LSTM cell structure.

The first layer is the sigmoidal layer of the forget gate layer. It calculates the factors for the memory vector components. In other words, a gate for information from the C_{t-1} vector is formed. This sigmoidal gate can be expressed by the formula:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

(7)

Second and third layers
Their linear combination allows exact calculation of the information that should be stored in the new state of the memory cell.

$$C_t = f_t * C_{t-1} + i_t * C'_t \quad (8)$$

1. Input layer gate
Determines the information that should be updated.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \quad (9)$$

2. tanh-layer
Builds a new vector of C'_t values

$$C'_t = \tanh(W_C * [h_{t-1}, x_t] + b_C) \quad (10)$$

The fourth layer is the output gate layer.

1. The sigmoidal layer defines the information needed for the output.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \quad (11)$$

2. The tanh-layer “spreads” the memory vector in the range from -1 to 1 and multiplies itself at the output from the sigmoidal layer, thus forming the output information:

$$h_t = o_t * \tanh(C_t) \quad (12)$$

In the world of artificial intelligence, the variability of LSTM networks is highly developed, however, the general idea and principles of working with data are traced in all LSTM networks. In many works, modules with slightly noticeable corrections are used, which should be considered separately for each case, and there are usually no significant differences in implementations. Unlike traditional recurrent modules that overwrite the memory state at each iteration, LSTM networks are designed to decide whether certain data is needed. Thus, LSTM modules are able to reveal essential information when processing sufficiently long time intervals and sequences.

1.4 Building a network of predictions

The central element of the software solution is the LSTM recurrent neural network. When working with a neural network, we will operate with data from logs. In each specific subnet (*rt*, *cat*, *sapevent* [*x*]), an element from the log will be supplied as the *x* parameter to the algorithm input. The operation of this network can be divided into two stages - training and testing.

1.4.1 Data preparation

First of all, to facilitate the network operation, it is worthwhile to remove runouts (strong deviations in the training sample conditioned by rare actions that will lower the network quality) and smooth the graph discreteness using the Simple Moving Average method (SMA) with a pace of 3:

$$x_{i+3} = \frac{x_i + x_{i+1} + x_{i+2}}{3} \quad (13)$$

In order to avoid the network retraining, the training data is preliminarily divided into chains of 5 using the sliding window method and mixed, a similar procedure occurs with the validation sampling. Thus, the network will learn to predict every fifth action according to the four previous ones.

1.4.2 Structure

A sequence of actions represented by a feature vector is loaded to the network input. Further, all data is distributed in three LSTM units corresponding to each field (*rt*, *cat*, *sapevent* [*x*]), which have 256-dimensional output space. Then the data from the LSTM units pass into fully connected layers with a linear activation function for converting 256-dimensional space to one-dimensional (see Figure 5). Thus, the calculations of different entities occur in parallel and independently of each other.

Fig. 5. The structure of the constructed neural network.

1.4.3 Training

The training takes place using the back error propagation method with network deployment through time (BPTT - Backpropagation Through Time) [20]: at each iteration, the network assumption based on the chain of previous events and the internal state of the LSTM cell is compared with the correct answer in the training sequence. According to the comparison results, the coefficients for the conversion of weights and changes in the internal state are calculated.

Fig. 6. The graph of losses during training.

The training (Figure 6) lasts 500 iterations (experiments showed that on average the network shows the best results in the area of 100-200 epochs with a small spread from the extreme values depending on the training sample). In parallel with training, a threshold is calculated, which will determine whether the deviation is abnormal. Also, as in the case of smoothing the training sample by the SMA method, the sequence of network responses is smoothed, and then the maximum deviation between the predicted event and the event from the training sample is selected. In other words, the selection of Threshold is carried out in the way to reduce the number of errors of the second kind (the network recognizes events as abnormal, but in fact they are not).

1.4.4 Testing

Network predictions are tested at the best epoch, the choice of which was based on the correlation values (Figure 7) of the prediction line at the training and at the validation sample.

Fig. 7. The graph of true and predict correlation (blue - for training, green - for validation).

The choice of the best epoch is performed as follows: starting from the first epoch, the correlation between the prediction and the correct answers is considered, if it is greater than the current best epoch, then the correlation in the validation sample is compared and if it is also better, then this epoch is recorded as the best and its weights are stored in memory; if the correlation value is slightly less than in the previous epoch and their difference does not exceed 0.000001, then if the percentage of the number of false positives of the anomaly detector is less, the current epoch becomes the best one.

2 Conclusion

In order to maintain a recurrent neural network spry, it should be supplied with new data for training. Accordingly, it is worth providing for some mechanisms for further neural network training.

Since the working efficiency of the long-short-term memory is related to the way of encoding the information provided to the network, it should be taken into account how much the statistical information on the use of programs for the user changes:

- If the correlation is within the normal range, then the network can be retrained while maintaining the old state and using new data previously processed by the analyst for the presence of abnormalities.

- If the correlation between the months is extremely small, then the coding knowledge base should be updated in accordance with the new data, the coding tree should be calculated again and the network should be trained from scratch using updated events, which are also processed by the analyst for the presence of abnormalities.

References

1. Y. Chen, J.S. Chou, *The International Conference on Digital Security and Forensics (DigitalSec2014). The Society of Digital Information and Wireless Communication*, 56-67 (2014)
2. K. Kolcio, L. Breger, P. Zetocha, *2014 IEEE Aerospace Conference – IEEE*, 1-14 (2014)
3. P. Esling, C. Agon, *ACM Computing Surveys (CSUR)* **45(1)**, 12 (2012)
4. Y. Li, Y. Wang, *International Journal of Digital Content Technology and its Applications*, *Advanced Institute of Convergence Information Technology* **6(5)**, 25-33 (2012)
5. V. Chandola, A. Banerjee, V. Kumar, *IEEE Transactions on Knowledge and Data Engineering* **24(5)**, 823-839 (2012)
6. S.M. Kovalev, A.V. Sukhanov, V. Styskala, *Proceedings of 1st European- Middle Asian Conference on Computer Modelling* (2015)
7. *Long Short-Term Memory*, <https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html>
8. *Understanding LSTM Networks*, <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>
9. *SAP Company Information*, <https://www.sap.com/corporate/en/company.html>
10. *SAP Help Portal*, <http://help.sap.com>
11. *SAP Enterprise Threat Detection*, <https://www.sap.com/cis/products/enterprise-threat-detection.product-capabilities.html>
12. *SAP Cyber Security in Figures. Global Threat Report*, <https://erpscan.com/wp-content/uploads/publications/Sap-Cyber-Threat-Report.pdf>
13. *Vanishing Gradient Problem*, https://en.wikipedia.org/wiki/Vanishing_gradient_problem
14. *Detecting Security threats in the Router using Computational Intelligence*, <https://arxiv.org/ftp/arxiv/papers/1005/1005.0967.pdf>
15. *Automated detection of smuggled high-risk security threats using Deep Learning*, <https://arxiv.org/pdf/1609.02805.pdf>