Grzegorz LITAWA[1], **Mateusz ŻELASKO**[2], **Julia LITAWA**[3]

[1] University of Applied Sciences in Nowy Sącz, Faculty of Engineering Sciences, Zamenhofa 1a, 33-300 Nowy Sącz, e-mail: glitawa@ans-ns.edu.pl
[2] University of Applied Sciences in Nowy Sącz, Faculty of Engineering Sciences, Zamenhofa 1a, 33-300 Nowy Sącz, email: mat.zelasko@gmail.com
[3] University of Applied Sciences in Nowy Sącz, Faculty of Engineering Sciences, Zamenhofa 1a, 33-300 Nowy Sącz, email: jula.litawa@gmail.com

Detection and analysis of network security threats using artificial intelligence

Abstract

This article describes ways of detecting and analysing major security threats in network. Our research is based on a neural network created in Python language with the use of TensorFlow. With the help of artificial intelligence, we can analyse language for the presence of disinformation and propaganda. We have used BERT (Bidirectional Encoder Representations from Transformers) model, in order to catch fake, malicious and suspicious content. Basing on the BERT model, other models were created, for example HerBERT, which was trained to understand Polish. Furthermore, we have availed ourselves of DNS analysis, so we can localize IP addresses of Internet domains. In this work, we will discuss these methods of protecting information space.

Keywords: machine learning, language analysis, protecting information space, LSTM, Keras. BERT, DNS, NLP.

Wykrywanie i analiza zagrożeń bezpieczeństwa sieci przy użyciu sztucznej inteligencji

Streszczenie

W artykule opisano metody wykrywania i analizy głównych zagrożeń dla bezpieczeństwa sieci. Nasze badania opierają się na sieci neuronowej stworzonej w języku Python przy użyciu biblioteki TensorFlow. Dzięki sztucznej inteligencji jesteśmy w stanie analizować język pod kątem obecności dezinformacji i propagandy. Wykorzystaliśmy model BERT (Bidirectional Encoder Representations from Transformers), aby wykrywać fałszywe, złośliwe i podejrzane treści. Na bazie modelu BERT stworzono inne modele, takie jak HerBERT, który został przeszkolony do rozumienia języka polskiego. Ponadto skorzystaliśmy z analizy DNS, dzięki czemu jesteśmy w stanie zlokalizować adresy IP domen internetowych. W pracy tej omówimy te metody ochrony przestrzeni informacyjnej.

Słowa kluczowe: uczenie maszynowe, analiza języka, ochrony przestrzeni informacyjnej, LSTM, Keras, BERT, DNS, NLP.

1. Introduction

Natural Language Processing (NLP) has become a key field in the area of artificial intelligence, and the Bidirectional Encoder Representations from Transformers (BERT) model has played a significant role in its development. Natural language analysis is the field concerned with understanding, processing, and generating human language by computers. This is important due to the increasing amount of text data available online, such as articles, reviews, social media and much more. Artificial intelligence, especially NLP models, helps in the automatic understanding and processing of these texts (Géron, 2020; Brownlee, 2020).

The BERT model developed by Google is one of the most important innovations in the field of NLP. This based on the Transformers architecture model has gained huge popularity thanks to its ability to understand context and relationships between words in sentences. Its "bidirectionality" means it is able to analyze both the previous and next words in a sentence, leading to more precise natural language processing (Srebrovic et al., 2020; Bezliudnyi et al., 2023).

The BERT model is used in various fields such as sentiment analysis. It helps in understanding whether a given text is positive, negative or neutral, which is useful in monitoring content transmitted on networks. Search engines improve the relevancy of search results by taking into account the meaning of keywords in context. Machine translation enables more precise translations between different languages. Fake news detection helpfuls in identifying disinformation and fake news on social media and the internet. Natural language analysis using artificial intelligence and the BERT model has a huge impact on the way we process and reason about texts. With this tool, we are able to extract valuable information from the text data ocean, which is applicable in many fields, from business to scientific research. The development of NLP and models like BERT promises even more exciting applications in the future.

2. Language analysis

Nowadays, we have to face threats connected with information war, because they affect our world in many ways. We have to give end-user a chance to protect from it. Artificial intelligence models can be used in tasks in the range of natural language processing – including search, chatbots, sentiment analysis, and autocomplete. We have utilized it in detecting propaganda. The 2016 US presidential elections resulted in collecting a big dataset of Twitter posts marked as Russian propaganda, which we have used in model training. In 2023 there exist a lot of pre-trained models suitable for tasks like this. Propaganda comes in many forms, but it can be recognized by its persuasive function, sizable target audience, the representation of a specific group's agenda, and the use of faulty reasoning and/or emotional appeals (Miller, 1939). Since propaganda is conveyed through the use of a number of techniques, their detection allows for a deeper analysis at the paragraph and the sentence level that goes beyond a single document-level judgment on whether a text is propagandistic. In our research, we have chosen BERT model, because it can be fine-tuned in order to fit the given task of propaganda recognition (e.g. repetition, red herring) (Da San Martino et al., 2019; Bezliudnyi et al., 2023).

3. Python program for Propaganda classification using TensorFlow

Splitting data into training, validation, and testing sets is a common practice in machine learning. Its purpose is to evaluate and optimize the performance of a model. The training set is used to train the model by adjusting its parameters based on the examples in the set. It aims to help the model learn the underlying patterns in the data. The validation set is used to evaluate the performance of the model during training and to tune the hyperparameters. Hyperparameters are parameters that cannot be learned from the data. They are set before training (e.g. learning rate, number of layers and number of neurons). This set is used to determine the optimal values for these hyperparameters. The purpose of using the testing set is to evaluate the final performance of the model after it has been trained and optimized on the training and validation sets. It helps to

estimate the generalization error of the model, which is the error rate on new, unseen examples. The testing set is essential to assess the model's ability to generalize to new data and to compare the performance of different models (Albon, 2018).

```
[ ]: import random
     random_state = random.randint(0, 9999)

     train, test = train_test_split(dataset, test_size=0.1,␣
      ↪random_state=random_state)
     train, val = train_test_split(train, test_size=0.2, random_state=random_state)
```

Figure 1. Split the data into training, validation, and test sets

```
[ ]: train['label'] = train['label'].astype('int32')
     val['label'] = val['label'].astype('int32')
     test['label'] = test['label'].astype('int32')
```

Figure 2. Convert label column to integers

4. Define the BERT layer

We load Keras layer from TensorFlow Hub for preprocessing the input text data. This layer is based on the BERT architecture and is specifically designed for English language text data that is uncased. Second layer loads another Keras layer from TensorFlow Hub that contains a pre-trained BERT model for English language text data. The trainable=True argument makes the BERT layer trainable during the fine-tuning process (Brownlee, 2020).

```
[ ]: bert_preprocess_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/
      ↪bert_en_uncased_preprocess/3")
     bert_layer = hub.KerasLayer('https://tfhub.dev/tensorflow/
      ↪bert_en_uncased_L-24_H-1024_A-16/4', trainable=True)
     # CHANGE IT FOR SOMETHING MORE SUFFICIENT !!!
```

Figure 3. Define the BERT layer

An LSTM cell can be considered in two ways. The first is to treat it as a black box. In this case, the state of the cell is divided into two vectors: $h_{(t)}$ – corresponds to the short-term state of the cell and $C_{(t)}$ – corresponds to the long-term state of the cell. The second way is to define the long-term state of $C_{(t-1)}$ starting on the left side of the cell and ending on the right side of it. Long-term state, it passes the forget gate and changes its value, then sums up with the entry gate and is passed on without any modification. The long-term state $C_{(t)}$ is copied and passed through the *tanh* function (hyperbolic tangent), the result of this operation is filtered by the output gate and the state $h_{(t)}$ is created – short-term. This state is equal to the output of the cell for this time step (Brownlee, 2020; Essien et al., 2020).

5. Add an LSTM layer with 64 units

We use the LSTM layer to further process the contextualized word embeddings generated by a BERT model. LSTM model is a special type of RNN architecture that is capable of learning long-term dependencies because the memory, which is implemented by the classical RNN model, is short-lived – at each training step, the information in the memory is combined with new information and is completely overwritten after a few iterations (Géron, 2020; Brownlee, 2020; Essien et al., 2020). Mixing LSTM and BERT layers gave us the even better performance of this state-of-art model.

```
[ ]: lstm_output = tf.keras.layers.LSTM(64)(bert_output['sequence_output'])
```

Figure 4. Define the LSTM layer with 64 units

Then, we need to batch the datasets. Batching is the process of splitting a dataset into smaller subsets, called batches or mini-batches, to be processed and trained on a model in parallel. Dataset batching is a technique used in deep learning and other machine learning applications to train models more efficiently. Instead of training the model on the entire dataset at once, the dataset is divided into smaller batches or mini--batches, and the model is trained on each batch separately. There are several benefits of batching. Model is processing smaller batches of data at a time. Regardful to it model can be trained faster. Furthermore, this process reduces amount of memory required to train the model. In addition, batching is helpful in improving the generalization of the model, because it prevents overfitting on individual examples (Loukas, 2020).

This model after training can be connected to the end user interface as for example browser extension which would analyse text and mark for the user probably harmful information. If detected propaganda text contains a hyperlink, it could be collected and then used for online training of the DNS analyser described below.

6. DNS domain classification

Domain Name System is a protocol, which associates IP address with more user--friendly domain names. It is composed of two or more parts, separated by dots. These parts are also called labels or segments. This system is hierarchical. The rightmost label represents the top-level domain (TLD) while the leftmost label represents the specific domain. It is very important for Internet because it is used by many internet services, even those malicious. DNS plays a significant role in identifying and blocking access to domains, which might be associated with malicious activity. We have created a neural network model which distinguish between benign and malicious DNS domain. As the DNS domains are not very complicated the preprocessing and the model used are not sophisticated, just character-level text tokenization and the basic Deep Neural Network model (Demertzis et al., 2019).

7. Network traffic analisys

The TextVectorization layer in TensorFlow is a preprocessing layer that can be used to vectorize text data. Vectorizing is the process of converting data into a numerical vector or array format that can be processed by a machine learning algorithm. In machine learning, the model input data needs to be in a numeric format, so it can be processed by the algorithms that perform computations on the data (Géron, 2020).

On the lower level of internet communications techniques lays the flow of IP packets between internet hosts. In modern times most of the network traffic is encrypted by protocols like HTTPS or SSH, so it is pointless to look for information related to propaganda or misinformation by itself in it, but it can be used to detect anomalies in network traffic which may indicate a compromised host. It is important for information space as it may allow threat actors to acquire credentials to social media accounts which could be used then for the purpose of spreading propaganda and misinformation. Also, if compromised man has access to some sensitive data, it can be used as a fuel for psychological warfare. This scenario occurred in Poland in 2021 when the head of the

chancellery of Poland's prime minister had his email hacked, which led to the leak of sensitive documents and emails online. For training and testing there was used USTC--TFC2016 dataset which consists of samples of benign and malicious network traffic from different sources. The dataset was also pre-processed by mapping IP addresses into random, so the model would not learn specific addresses as malicious or benign, which can vary. The important thing to notice is the fact that cybersecurity is a constant arms race – threats are constantly changing, so it is important to ensure that the model deployed in the wild would be trained using the most actual samples.

Analysing of network traffic is a difficult task that can be solved in many ways. We're proposing a combination of two different approaches which differs in data preprocessing but are using the same neural network model.

We concentrate on Transmission Control Protocol and User Datagram Protocol flows, splitting them and analysing only two first packets. This naive approach bases on the fact that in network flow most of datagram fields do not change, so only a small probe should be sufficient to classify flow as benign or malicious. As the most significant advantage, it is relatively small computational complexity of preprocessing data, so this approach could be utilized in real-time network traffic analysis. We are splitting network traffic into different flows using IP addresses and ports. IP addresses are randomized so the model will not be dependent on specific subnetwork. Then the flows are converted into 24-bit RGB bitmaps with resolution 24x42 which is suitable for two Ethernet frames. If (and this scenario happens in most situations) packets are not maximum size, the rest of bitmap is padded with zeros, resulting in black pixels. Then two first packets of every flow are converted into 24-bit RGB bitmaps with resolution 24x42 (Demertzis et al., 2019).

8. DNS domain classification

Data preprocessing incorporates converting network traffic data into images in order to make use of the methods used in image analysis – probably the most studied case of using artificial intelligence. Convolutional neural networks (CNNs) are a commonly used type of deep learning algorithm in image analysis, and they have been found to be particularly effective in processing images created from network traffic data. This is because CNNs are designed to identify and extract meaningful patterns and features from images through a series of convolutional and pooling layers. By converting network traffic data into images, data preprocessing can enable the use of CNNs for analysing and classifying network traffic. The rescaling layer in TensorFlow Keras can be used to scale the input data to be between 0 and 1 by dividing each pixel value by the maximum possible value, which is 2^24 – 1. By applying this rescaling operation to the input data, the pixel values are normalized, and the model is more capable to learn the important features and patterns in the data. Additionally, normalization helps to prevent the model from being overly sensitive to the range and magnitude of the input data, which can improve the model's accuracy and generalization ability (Livieris, 2020).

The presented models can be used for comprehensive user protection, for example, in conjunction with a web browser interface. They can also be used to detect and analyse domains from which potentially harmful content is being sent, in order to take further steps to protect users, messengers, websites, or entire computer networks (Chollet, 2019).

9. Conclusions

During our research, we have created programs with use of artificial intelligence, which can help use face cyber security threats. The first one, which has been described in more specific way, is analysing text. It can be helpful in protecting people from disinformation. We haven't concentrated on the code of second program, because of it obvious drawbacks, that impact its accuracy. In further research, we can improve it, by for example, giving the model more examples or we can fit it better. As a result model would be more adjusted to protect our devices from malware and malicious domains and protect users, messengers, websites, or entire computer networks.

References

Ai, Y., Li, Z., Gan, M., Zhang, Y., Yu, D., Chen, W., Ju, Y. (2019). A deep learning approach on short-term spatiotemporal distribution forecasting of dockless bike-sharing system. *Neural Comput Appl, 31*(5), 1665-1667.

Albon, C. (2018). *Uczenie maszynowe w Pythonie. Receptury*. Gliwice: Helion.

Bezliudnyi, Y., Shymkovych, V., Kravets, P., Novatsky, A., Shymkovych, L. (2023). *Pro-Russian propaganda recognition and analytics system based on text classification model and statistical data processing methods*. Retrieved from: https://ela.kpi.ua/bitstream/123456789/55707/1/278923-643037-1-10-20230510.pdf.

Brownlee, J. (2020). *How to Develop LSTM Models for Time Series Forecasting*. Retrieved from: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/.

Brownlee, J. (2020). *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. Retrieved from: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

Chollet, F. (2019). *Deep Learning. Praca z językiem Python i biblioteką Keras*. Gliwice: Helion.

Da San Martino, G., Seunghak, Y., Barron-Cedeno, A., Petrov, R., Nakov, P. (2019). *Fine--Grained Analysis of Propaganda in News Articles*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (pp. 5636-5646). Hong Kong.

Demertzis, K. Iliadis, L. Bougoudis, I. (2019). *Gryphon: a semi-supervised anomaly detection system based on one-class evolving spiking neural network*. Neural Comput Appl.

Essien, A. Giannettic C. (2020). A Deep Learning Model for Smart Manufacturing Using Convolutional LSTM Neural Network Autoencoders. *IEEE Transactions on Industrial Informatics PP, 99*, 6069-6078.

Géron, A. (2020). *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*. Gliwice: Helion.

Livieris, I.E. (2020). A CNN–LSTM model for gold price time-series forecasting. *Neural Computing and Applications, 32*, 17351-17360.

Loukas, S. (2020). *Time-Series Forecasting: Predicting Stock Prices Using An LSTM Model*. Retrieved from: https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f.

Srebrovic, R., Yonamine, J. (2020). *Leveraging the BERT algorithm for Patents with TensorFlow and BigQuery*. Retrieved from: https://services.google.com/fh/files/blogs/bert_for_patents_white_paper.pdf.