ISSN 2007-9737

Prerequisites of in-Context Learning for Transformers on Queries

Bulat Shkanov[1], Mikhail Alexandrov[2,3,*]

[1] Gaidar's Institute for Economic Policy,
Russia

[2] RANEPA,
Russia

[3] FRUCT,
Finland

bulat.shkanov@mail.ru, malexandrov@mail.ru, malexandrov.uab@gmail.com

Abstract. Pre-trained generative transformers (GPT) with their 200+ billion parameters have already demonstrated their ability to successfully solve a wide range of text-related problems without the need for additional task-specific training. However, it has been observed that solution quality can be significantly improved for certain queries that reflect task formulation and conditions. It indicates that the transformer is further trained based on the query context, and the aim of this study is to show why GPT transformers enable to do it. To this end, the article jointly considers: elements of transformer architecture (data compressors and sentiment neurons), elements of the user interface with transformers (zero-shot and few-shot prompts), and text processing procedures (arithmetic coding and minimum description length). The authors attempt to provide a theoretical justification for the convergence of the sequential fine-tuning process using Hoeffding's inequality. The study presents experimental results demonstrating GPT transformers' capabilities for in-context learning. This confirms their potential for further development in natural language processing technologies.

Keywords. Data compressors, sentiment neurons, in-context learning, zero-shot learning, few-shot learning

1 Introduction

1.1 Motivation

In recent years, large language models and their implementation as GPT transformers have become one of the most discussed and in-demand technologies in artificial intelligence. They have firmly entered everyday life and are used in various fields as personal assistants, improving search and enabling creative content generation. This success is partly due to their ability to learn from the context of queries.

To understand why modern large language models, possess such capabilities, we must consider the foundational concepts underlying them and experimental results.

2 Fundamental Concepts

We divide the foundational concepts that explain transformers' ability for in-context learning into three groups.

2.1 Regarding the Architecture and Functioning of the Transformer

- Data compressor as a tool for information compression. This tool not only reduces information but also structures it. This allows generative models to effectively generalize knowledge and use it to solve new tasks.
- Sentiment neurons. These are parts of the neural network activated in the presence of positive or negative sentiments in a text. These neurons exemplify how models can extract and represent complex emotional nuances of the text, forming part of their data compression capabilities.

2.1.1 Regarding the Training Process

- In-context learning is a method where a language model learns to solve a task not by traditional parameter updates, but by being provided with examples or instructions in the query context. This means the model can adapt to a task by considering the context without needing a separate training phase.
- Zero-shot queries involve interacting with a GPT transformer where the model solves a task without examples related to that task. The model uses its knowledge and generalization ability to perform a new task based on contextual information or instructions.
- Few-shot queries involve providing the model with a few training examples. This approach is especially useful when training data is limited, but high accuracy and knowledge transfer are required.

2.1.2 Regarding Processing Procedures

- Arithmetic coding. A method of data compression that encodes symbol sequences using fractional values. The core idea is to represent the entire symbol sequence as a single number within a specific interval rather than encoding each symbol separately.
- Minimum Description Length (MDL) is a model selection method based on the idea that the best model for data description is the one that minimizes the total description length. It includes the length of the model description (how well the model "compresses" the data by explaining its patterns) and the length of the data description (e.g., number of parameters, their precision, etc.). In other words, MDL aims to balance model complexity and its accuracy in explaining observed data.

2.2 Problem Statement

Researchers attempting to explain GPT models' in-context learning abilities typically consider the above foundational concepts in isolation. Unlike them, the authors believe these abilities arise from the integration of these concepts. This integrative approach is explored in this work.

The article is structured as follows. Section 2 describes the transformer architecture and training method. Section 3 offers a mathematical model explaining learning process convergence. Sections 4 and 5 detail compression models and sentiment neuron tuning. Section 6 presents zero-shot and few-shot learning concepts and experimental results. Section 7 contains the conclusions of the study.

3 Transformer and the Training Process

3.1 Transformer Architecture

The transformer architecture is one of the most significant innovations in deep learning, introduced in 2017 in the paper "Attention is all you need" [1]. This work revolutionized the processing of sequential data such as text. It introduced the main components of transformer architecture, described below.

3.1.1 Self-Attention Mechanism

Self-attention mechanism allows the transformer to analyze the entire input text (or other data sequences, e.g., time series) as a whole and determine which parts of the sequence are most important for processing each individual element. This is one of the key properties of the transformer that distinguishes it from other deep learning architectures, such as recurrent neural networks (RNNs), which process text sequentially.

Self-attention uses three elements represented by their matrices:

- Query, it is representation of the current word being processed.
- Key, it is representations of all words in the sequence.
- Value, it is contextual representations of the words.

The model computes the similarity between the query and the keys to determine the weight of each value. It then aggregates the results to create a new representation of the current word considering all other words.

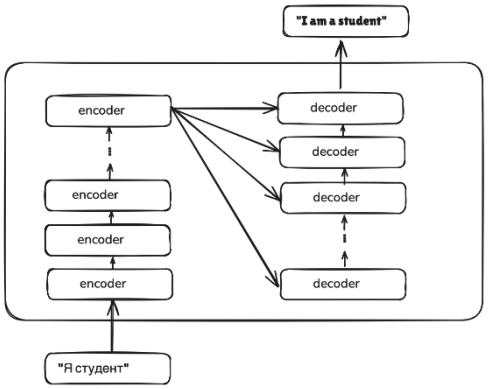

Fig.1. Simplified transformer architecture

3.1.2 Multi-Head Attention

Multi-head attention allows the transformer to learn various aspects of word dependencies. Instead of using a single attention mechanism, the transformer applies multiple "heads" of attention. Each "head" computes its own attention and then the results are combined. This enables the model to capture more complex dependency patterns between words.

3.1.3 Encoder and Decoder

The self-attention mechanism and the layers of connected neurons are included in two main components of the transformer:

- Encoder is used for processing input data. Each of its layers includes attention mechanisms and weights of fully connected neurons (feedforward layers). Each word in the sequence is transformed using self-attention, and the result is passed to the next layers.
- Decoder is used for generating the output sequence based on the encoded information. The decoder also uses attention mechanisms, but unlike the encoder, it additionally applies attention to the original sequence (encoder-decoder attention). This allows it to consider both its own previous outputs and the encoded input data.

Figure 1 shows here these elements in a very simplified way in the process of translation (see, 2.2 below).

3.1.4 Normalization and Residual Connections

To improve learning and prevent gradient vanishing, residual connections are used at each layer of the transformer. These connections:

- Add an original input data to the layer's outputs.
- Introduce a layer normalization step to stabilize training.

3.1.5 Feed-Forward Layers

After each attention layer, standard feedforward neural networks are applied independently to each sequence element. This helps to improve data representation at each processing step.

3.1.6 Positional Encoding

As noted above, transformers do not process data sequentially like recurrent networks. Therefore, they lack a built-in mechanism for capturing word order in a sequence. To solve it, positional encoding is used, namely each word in the text is assigned a position number in the sequence. This allows the model to account for element order.

3.1.7 Conclusions on Transformer Architecture

Transformers offer several advantages over RNNs/LSTMs for sequence processing:

- The entire sequence can be processed in parallel, whereas RNNs process step- by- step.
- Transformers can be trained on very large datasets by scaling width and depth (as seen in language models with billions of parameters), while training such deep LSTMs is extremely difficult.
- Self-attention more easily captures long-range dependencies.

However, transformers have two disadvantages compared to classical linear models:

- They require large amounts of data and computations.
- When trained on small datasets, transformers often lose generalization ability and overfit more quickly.

Table 1. Annotation fragment from the example

Token from Vocabulary/Position	1	2	3
«a»	0	0	1
«am»	0	1	0
«I»	1	0	0
«thanks»	0	0	0
«student»	0	0	0

Table 2. Transformer prediction fragment from the example

Token from Vocabulary/Position	1	2	3
«a»	0,01	0,01	0,99
«am»	0,02	0,80	0,001
«I»	0,93	0,10	0,001
«thanks»	0,01	0,05	0,001
«student»	0,03	0,001	0,002

- The transformer architecture is the foundation for many modern language models such as GPT, BERT, and their successors. It has opened new possibilities for the advancement of artificial intelligence.

3.2 Transformer Training

The transformer architecture, developed in 2017 by researchers from Google, was originally created for machine translation tasks. Previously, recurrent neural networks were used for such tasks, but they had several drawbacks that hindered effective use. A breakthrough occurred in mid-2017 with the publication of the article "Attention is All You Need" (Vaswani et al., 2017).

When we input Russian text into the model, it undergoes a certain process, and as output we receive a translation in English. This task belongs to the class of supervised learning problems, and training such a model requires collecting a huge number of text pairs, e.g., Russian-English pairs. This involves engaging many people to form these pairs, organizing the annotation process, paying the translators, and setting up the labeling system.

Let's assume we know the translation of a Russian phrase into English: «Я студент» => «I am a student». Table 1 shows a fragment of annotation for training a transformer to solve this machine translation task.

For the first position of the phrase in Russian, we must predict the token or word "I", then "am" and so on. We don't need to map the target translation to specific positions directly: we just write one long sentence that is the correct translation. The transformer algorithm uses a tokenization mechanism to break this sentence into positions, but the original translation is still required.

Once the model is trained for translation, it will make predictions based on dictionary probabilities. Table 2 shows a fragment of transformer predictions for the same translation task. The labeling is again binary: 1 or 0.

For example, if the first token is supposed to be "I" a well-trained model will likely predict "I" as the first token with 93% confidence. Then, when asked what token should come second, it predicts "am" with 80% probability. This is an example of how supervised learning is performed.

Even though the model predicts one word after another, this is still supervised learning, and we can provide a mathematical justification for why this mechanism works.

4 Hoeffding's Inequality and the Learning Process

Hoeffding's inequality [2] is a theorem in probability theory used to estimate the likelihood that the sample mean of a random variable significantly deviates from its expected value. In the context of supervised machine learning, this inequality provides a confidence bound on the convergence of training algorithms, indicating the probability that the empirical mean (computed from a sample) significantly differs from the true mean.

In machine learning and mathematical statistics, error is defined as the deviation of the model's prediction from the true value. In this context, two types of errors are distinguished:

- True error (generalization error). This is the expected value of the model's loss function over the entire (unknown) data distribution. It reflects the model's real ability to generalize

the knowledge it learned from the training sample.

- Empirical error (training error, empirical risk). This is the average error value calculated over the finite training dataset. In other words, the empirical error shows how well the model performs on the data it was trained on.

Since the training set is only a finite sample from the full distribution, the empirical error differs from the true error. As the sample size increases, the empirical error should converge to the true error. This is indirectly reflected in the law of large numbers.

Hoeffding's inequality for a sample $X_1, X_2, ..., X_n$ of independent and identically distributed random variables in the interval [a,b] gives the following bound for any deviation $t>0$:

$$P(|\bar{X} - E[\bar{X}]| \geq t) \leq 2e^{-\frac{2nt^2}{(b-a)^2}}, \quad (1)$$

where $E[\bar{X}]$ is the expected value of the random variable $\bar{X}$.

Hoeffding's inequality shows that as the training sample of size n increases, the probability of a large deviation of empirical error from true error tends to zero. This guarantees that the training algorithm will converge to the true parameters if the training dataset is sufficiently large. This conclusion underscores the significance of Hoeffding's inequality for supervised learning algorithms.

5 Data Compressor

5.1 Unsupervised vs. Supervised Learning

When considering transformer learning from the context of queries, it is natural to raise the following questions:

- What happens to the model when we have no pre-prepared training datasets, that is, when we use unsupervised learning instead of supervised learning?
- Is there a mathematical framework for this case that could help to answer the question?

Unsupervised learning is not limited to natural language processing tasks. In computer vision tasks, most models are trained using unsupervised learning.

Here are some common situations where unsupervised learning is applied:

- When data labeling is difficult. This could be due to the large volume of data or the lack of experts capable for performing high- quality labeling.
- When we have a dataset of unlabeled data and we intentionally avoid labeling it to train a model that is more robust and capable of generalization.

5.2 Arithmetic Coding

Arithmetic coding, as a data compression method, is closely related to machine learning. The following processing procedures reflect this connection:

- Data storage. In machine learning, large volumes of data are often involved, and compression can significantly reduce storage and transmission requirements. Arithmetic coding allows data to be compressed with minimal loss, which is especially important for storing training and test datasets.
- Entropy coding. Arithmetic coding is an example of entropy coding used to minimize the average code length. In machine learning, - especially in tasks involving probabilistic modeling (e.g., Hidden Markov Models, Bayesian Networks), - entropy coding helps to optimize the representation of probability distributions and improve model efficiency.
- Generative Adversarial Networks (GANs). In GANs, arithmetic coding helps to create compact data representations, enhancing the model's ability to generate new data.
- Security and data protection. In tasks related to data security and protection, arithmetic coding can be used both for compression and for encryption. This is particularly important for confidential data used in machine learning.

The performance of large language model training algorithms can be evaluated using the parameter of entropy, borrowed from information theory. Information entropy measures the

uncertainty in a system, particularly the predictability of a primary alphabet symbol's occurrence. In our case, the primary alphabet consists of the symbols that the language model processes as input and output. We can estimate how likely or unlikely it is that the model will correctly predict a given sequence of words.

Let us consider two datasets: *X* and *Y*. Suppose we have a compressor that compresses data well. The compressor should use patterns from *X* to better compress *Y*. That is, if we have additional information from *X* that is relevant to *Y*, we can use it to better compress *Y*. The reverse is also true, namely, if we have additional information from *Y* relevant to *X*, we can use it to better compress *X*. This principle can be formally expressed by the inequality:

$$|C(concat(X,Y))| < |C(X)| + |C(Y)| + O(1), \quad (2)$$

where: C denotes the compression operation, and concat (concatenation) denotes the operation of combining two datasets.

This applies to prediction as well: if we use a sample that contains extractable useful patterns, the predictive model that captures these patterns should also reduce entropy on that sample.

5.3 Minimum Description Length

Minimum Description Length, MDL [3] is a parameter (or criterion) used to guide model selection based on a balance between model complexity and its ability to explain the data. This parameter is closely related to the concept of data compression and entropy coding. In machine learning, MDL helps to prevent overfitting.

The MDL principle states that the best model for describing a dataset is the one that minimizes the total description length of the model and the data. It includes:

- the length of the model description itself,
- the length of the data description as used by the model.

A model that efficiently compresses data also minimizes its entropy. At the same time:

- complex models reduce data uncertainty but increase the length of the model description.
- simple models reduce the model description length but increase data uncertainty.

The Minimum Description Length principle ensures the right balance between the two factors above.

6 Sentiment Neurons

Research into sentiment analysis mechanisms in large language models (LLMs) was initiated by OpenAI in 2017. The authors of the study "Learning to generate reviews and discovering sentiment" [4] proposed an approach for using sentiment neurons to generate textual reviews and detect sentiment in texts. The study tested two hypotheses:

- Hypothesis 1: Models trained on large volumes of textual data can effectively generate realistic text reviews.
- Hypothesis 2: During text analysis, the model can automatically detect sentiment in these texts and classify the texts.

The results of the experiments from that article are presented below.

6.1 Experiment 1

The experiment focused on training a model to generate texts. The researchers trained a large language model on a dataset of textual reviews. The model was configured to generate texts that mimic the style and content of real reviews. An unsupervised learning algorithm was used, which allowed the model to learn from a variety of text data without prior annotation. The experiment confirmed Hypothesis 1: the model generated texts that were stylistically and semantically close to actual reviews.

6.2 Experiment 2

This experiment focused on identifying sentiment neurons. The researchers analyzed neuron activations in the model to identify those responsible for sentiment analysis. They discovered that certain neurons were activated in the presence of positive or negative sentiments in

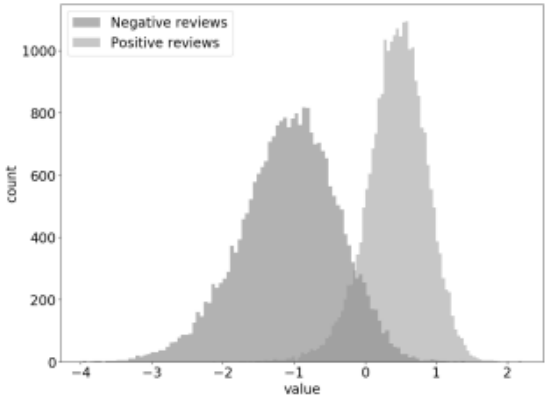

Fig. 2. Activation distribution of a neuron responsible for review sentiment

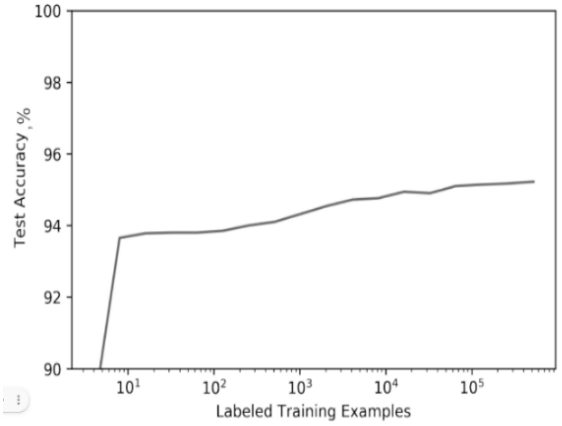

Fig. 3. Relationship between classification accuracy and the size of the training dataset for the Yelp reviews dataset

the text, while others were not. These active neurons were labeled “sentiment neurons.”

Figure 2 shows the activation level of one such sentiment neuron when analyzing a set of texts containing sentiments. The experiment confirmed Hypothesis 2: the model detected and classified sentiment in the texts.

6.3 Experiment 3

This experiment focused on sentiment classification. The model was tested on a dataset of reviews labeled as “positive” or “negative.” Activation of sentiment neurons was used to predict the overall tone of each review.

Figure 3 shows the classifier’s accuracy depending on the size of the training dataset. Accuracy reaches its maximum when fine-tuned on just 10 examples. This experiment once again confirmed Hypothesis 2: the model successfully detected and classified sentiments in texts and provided an overall assessment of text tone.

The research results discussed above [4] demonstrated that large language models can not only generate realistic texts but also automatically analyze sentiment in those texts. The identified sentiment neurons showed high accuracy in sentiment classification, opening new opportunities for applying such models in various domains.

7 Concept of Zero-Shot and Few-Shot Learning

In mid-2018, OpenAI released the first Generative Pretrained Transformer (GPT). The capabilities of the transformer were documented in the article “Improving language understanding by generative pre-training” [5]. This article provides a detailed examination of the concepts of zero-shot and few-shot learning, presents experimental examples, and discusses the results. We summarize these materials below.

7.1 Experiment 1

Experiment Focus: Zero-shot learning.

Goal: To test GPT-1's ability to perform various NLP tasks without any additional task-specific training. The model was evaluated on three tasks:

- Sentiment classification. Determining whether texts were positive or negative without training on labeled data. A binary classification task.
- Translation. Translating phrases between two languages based on prior pretraining knowledge. A machine translation task.
- Question answering. Answering questions based on general understanding of language and context.

Results for each task:

- GPT-1 demonstrated acceptable accuracy in classifying texts as positive or negative. However, it did not reach the performance of models specifically trained for the task.
- Translations produced by GPT-1 were reasonably accurate, though errors appeared in complex or context-heavy phrases.
- GPT-1 correctly answered most questions, showing an ability to extract relevant information from texts.

7.2 Experiment 2

Experiment Focus: Few-shot learning.

Goal: To assess the effectiveness of few-shot learning by adapting GPT-1 to new tasks using a minimal number of examples. The model was tested on the same three tasks:

- Sentiment classification. The model was trained using just a few labeled examples of positive and negative reviews.
- Translation. A few pairs of example sentences were provided in the source and target languages.
- Question answering. The model was given a few examples of questions and answers for training.

Results for each task:

- Classification accuracy improved significantly compared to zero-shot learning, especially with clearly defined examples.
- Translation quality increased, enabling the model to more accurately convey meaning.
- The model produced more precise answers to questions, demonstrating its ability to learn from examples.

7.3 Conclusions from the Experiments

- GPT-1 possesses zero-shot learning capabilities, allowing it to perform diverse tasks without requiring labeled data. However, accuracy and quality can vary depending on task complexity and context.

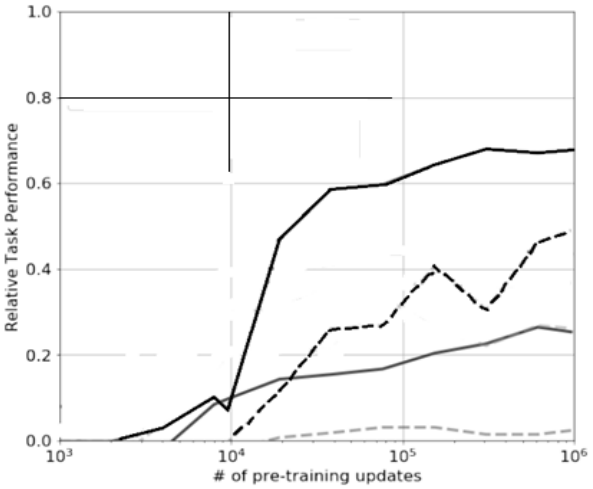

Fig. 4. Relationship between task performance and number of updates for GPT-1 and LSTM on sentiment and question answering tasks

- GPT-1 effectively utilizes few-shot learning, enabling it to adapt to new tasks with minimal data. This makes the model a flexible tool capable of rapidly adjusting to new problems with a limited number of examples.

Following Experiments 1 and 2, the authors of [5] conducted two additional zero-shot learning studies. Their goals were:

- To examine how GPT-1's task performance in zero-shot mode improves with more pretraining updates.
- To compare GPT-1's results with those of an alternative model-a recurrent neural network (LSTM)specifically trained for the same tasks.

Note: The term "update" here refers to a single gradient descent step in optimizing model parameters during pretraining. Thus, a greater number of updates corresponds to longer pretraining before zero-shot testing begins.

The graph in Figure 4 illustrates the results for two tasks:

- Sentiment classification.
- Question answering.

An LSTM neural network served as the baseline comparison model.

Graph interpretation:

- X-axis: Number of model updates during pretraining.
- Y-axis: Task accuracy (percentage of correct answers).
- Solid lines: GPT-1 performance. It is black for sentiment classification, and it is gray for question answering.
- Dashed lines: LSTM performance. It is black for sentiment classification, and it is gray for question answering.

As the graph shows, GPT-1's task accuracy steadily improves with more pretraining. Moreover, even in zero-shot mode, GPT-1 outperforms LSTM after a certain number of updates. This confirms the transformer architecture's ability to generalize knowledge and solve tasks based on text context alone.

8 Conclusion

In this study, we jointly examined the foundational concepts underlying the functioning of large models based on transformer architecture and demonstrated that their combination provides insight into the transformer's capacity for in-context learning.

Experiments with zero-shot and few-shot prompting in the GPT-1 release showed the model's ability to adapt to new tasks with a minimal number of training examples. This makes such models flexible tools for solving NLP tasks.

The releases of GPT-2, GPT-3, and GPT-4 already incorporate these capabilities, although their detailed review falls outside the scope of this study. However, we can note that the ability to learn from context also applies to other machine learning applications. For example, the Chronos model by Amazon [6], based on Google's T5 architecture [7], has been adapted to solve tasks in time series analysis and forecasting.

The authors hope that the material presented here will help uncover new prospects for both scientific research and practical applications of transformer-based models in various subject domains.

References

1. **Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017).** Attention Is All You Need. Doi: 10.48550/arXiv.1706.03762.
2. **Hoeffding, W. (1963).** Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association, Vol. 58, No. 301, pp. 13–30. Doi: 10.2307/2282952.
3. **Rissanen, J. (2005).** An Introduction to MDL principle. IEEE Trans. Information Theory, Vol. 46, No. 7, pp. 2537–2543.
4. **Radford, A., Jozefowicz, R., Sutskever, I. (2007).** Learning to Generate Reviews and Discovering Sentiment. Doi: 10.48550/arXiv.1704.01444.
5. **Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. (2018).** Improving Language Understanding by Generative Pre-Training. Computer Science, Linguistics.
6. **Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., Shchur, O., Rangapuram, S. S., Pineda Arango, S., Kapoor, S., Zschiegner, J., Maddix, D. C., Wang, H., Mahoney, M. W., Torkkola, K., Wilson, A. G., Bohlke-Schneider, M., Wang, Y. (2024).** Chronos: Learning the Language of Time Series. Doi: 10.48550/ arXiv.2403.07815.
7. **Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J. (2019).** Exploring the Limits of Transfer Learning with a Unified Text-To-Text Transformer. Doi: 10.48550/arXiv.1910. 10683.

Article received on 09/01/2025; accepted 24/07/2025.
**Corresponding author is Mikhail Alexandrov.*