Res Intell Manuf Assem, 2025, 3(1): 129-137
DOI: 10.25082/RIMA.2024.01.005

RESEARCH ARTICLE

Self-learning AI in Educational Research and Other Fields

Evgeniy Bryndin

Russian Committee for Standardization of Artificial Intelligence, Novosibirsk, Russia

Correspondence to: Evgeniy Bryndin, Russian Committee for Standardization of Artificial Intelligence, Novosibirsk, Russia; Email: bryndin15@yandex.org

Received: November 1, 2024;
Accepted: January 30, 2025;
Published: February 5, 2025.

Citation: Bryndin E. Self-learning AI in Educational Research and Other Fields. *Res Intell Manuf Assem*, 2025, **3**(1): 129-137.
https://doi.org/10.25082/RIMA.2024.01.005

Abstract: There are several areas where self-learning AI is actively used. Machine learning and deep learning allow you to identify patterns and improve performance. Algorithms such as neural networks can adapt and improve based on experience. Self-learning GPTs are used to dialogue with humans. Computer vision recognizes and classifies images. Recommender systems analyze user preferences and offer personalized solutions. Adaptive robotic industrial control systems can optimize processes by adapting to changing conditions and data. Self-learning intelligent systems help detect and respond to new threats and attacks by analyzing network traffic and user behavior. These technologies continue to evolve, opening up new research opportunities for students in the field of education. Self-learning AI helps programs learn, draw conclusions, and use them in the future. Programming languages do not consider algorithms as data. Programs do not have access to themselves. To learn, you need to change, and for this you need to have access to your own code. Then self-learning becomes possible. By generating the logic of self-learning algorithms, they can improve the program, it becomes different from its source code, and these changes must be saved. The interpreter of the algorithm improves the intelligence of the program, and it becomes the author of optimal solutions. The programming language of self-learning algorithms Author allows students to form the logic of self-learning algorithms to create intelligent systems that can help them in research activities. These systems are able to independently improve their skills and accuracy without explicit programming for each new type of task.

Keywords: Author programming language, intelligent self-learning program, research activity

1 Introduction to self-learning algorithms in the Author language

The file with the code "*.txt" has its duplicate "*.code". They form a module. When loading a module, the interpreter selects the file written later from the pair, the source code always remains untouched, and all changes that have occurred will be visible to the programmer in the duplicate file. The startup module must have an entry point (main) and to run it, you need to drag the file with the mouse to "author.exe". Any module can request additional loading of other modules (#include <me.txt>). Also, modules can be loaded and unloaded during code execution using the include("me.txt") and uninclude("me.txt") commands. Upon completion of the algorithm, the interpreter unloads the startup module. Each module has a counter for the number of modules that requested it, and if zero is reached, the module unloads its code into the duplicate file. The duplicate file of the module is rewritten only when at least one change has occurred in it (the module) or there was no duplicate at all. Thus, you can register the connection of modules as you wish, the module will always be in memory in one copy. You cannot connect yourself.

The module contains many functions. In the files, the program is presented as text. But when loading, the code is converted and developed to a dynamic structure of the algorithm scheme. For each function, a flow graph is created and the program gets access to it. When unloading the module, the algorithm scheme is converted back to program text. Variables in the language are not bound to a type, they do not have to be declared. A variable is also created when it is used in a construction for recording (a=0;). The value of any variable can have the following type: void, int, float, double, digit, char, string, interval, vector (set), set (unique set), map (associative array), program (algorithm tree), function (algorithm diagram), graf, module. When accessing for reading from an unknown variable, the "void" type is returned. The language also

has pointers, they are organized as a string with data for accessing a variable. A variable can contain a pointer to itself (p=&p;p=****p;).

The language can use a special operand “#”. It returns the “void” type and serves as a symbolic designation of a hidden block, which, like any other operand, can be replaced with a tree operator structure of any complexity. When the “void” value gets into the condition of a ternary operator, the interpreter, each time, at the moment of execution, is randomly determined between true and false. When a value of the “void” type gets into the condition, the current execution point of the algorithm, at this moment, is divided into two. One goes by truth, the other by false. In further interpretation, as necessary, the entire memory map is divided into two.

The division of the execution point can be done with a special function “rozpad()”. It accepts multiple values and returns each of them at its execution point. Thus, the expressions “if(#);” and “if(rozpad({1,0}));” are similar. The expression “n=rozpad({1,2,3})+rozpad({10,150});” will cause a split into six processes, with different values of the variable “n”: 11, 12, 13, 151, 152 and 153. To randomly decide between all the process variants, you need to call the function “define();”. In this way, you can reduce all the execution points to one.

The Author language allows storing ambiguous values in any variable. Thus, algorithms written in this language may at any time encounter ambiguity in the process of solving the assigned task. The operand “#” returns the value “void”. When it gets into a conditional branching structure, which any cycle consists of, the execution point of the algorithm is divided into two. One goes along the truth branch, the other along the lie branch. Each execution point has its own memory map, that is, relative to one interpretation position, all variables contain single-valued values. Thus, after executing the first line of the program, the process is divided into two. They both continue to execute the same program, but with different values of the variable “m”. Then each process is divided into two more. One pair goes through the array reversal command, and the other does not. Then all processes issue a report to their common array.

When you have to make a choice and choose one of the alternatives, you need to have one, clearly formulated criterion. Here is a universal and simple criterion. To indicate in the algorithm the need to select one version of the process, with its data, the language has a built-in function/command “define()”. It selects one execution point from the existing set at the time of its execution. How it works. The current process reaches the command “define()” and stops. Another, not stopped, process from the list of parallel processes becomes active. When all the processes in the list become stopped, this means that it is time to make a choice of one of them. One execution point from the list is selected, and all other processes are closed. The selected process is launched for further execution of the algorithm. Thus, regardless of the order of processing parallel processes, after executing the command “define()” only one execution point remains, objectively, guaranteed, with its own version of data.

In the Author language, there are data types “graf” and “program”. A graph is a set of nodes and links between them. For the “graf” type, a node exists only if it has at least one link. Each link has two node numbers, the corresponding value of the presence of input arrows and the name of the link. In addition to links between nodes, in a universal graph, there is also a link of a node with a marker. A marker is a text string. Each link with a marker has one node number, a link name and a marker name. There are also logical values about the presence of an input arrow in a node and in a marker. To create a variable of the “graf” type, you need to use the type casting operator. Next, we set the graph itself with a shape similar to an open envelope. The “G.getMARKER(0,#,#,0,#)” method searches among the links of a node with a marker according to the specified mask and returns an array of found elements. Each element is an array with the data requested in the mask. The search mask must contain at least one unknown link detail, which is indicated by the “#” symbol. In the Author language, the “#” symbol returns a “void” value, which can be obtained, for example, from a function without “return”. The array method “.export()” will convert nested arrays to a string. The “trace()” function prints a string to the screen. The graph method “G.getNET(4,#,#,#,#)” searches for connections between graph nodes taking into account the symmetry of the search and works similarly to searching for connections with markers.

To clearly demonstrate the capabilities of the fundamental type “program”, the following example. The function algorithm line can specify the terminal branch of recursion. Then the variables can access the function in which it is located. Then the variables receive a unique identifier of the nodes in the (block) of the algorithm scheme, into which the function has turned at the time of execution. The last node is above the zero node. Next, we get copies of the command, which is in the last node of the algorithm scheme. Actually, the “command” is an

object of the “program” type. It is a tree of operators and function calls with variables and constants. Any command can be either converted to text or executed. Using a method like “command.getSub({})” you can copy a branch from a tree by specifying an access path, in this case the entire tree will be copied. The “PROGRAM()” function converts a program text line to the “program” type.

2 Approaches and methods for writing intelligent self-learning programs

Creating self-learning programs requires a combination of theoretical knowledge and practical skills. It is important to choose the right approaches and methods depending on the specific task, available data and project goals.

2.1 Methods and techniques

(1) Data Preprocessing: Involves data cleaning, normalization, encoding categorical variables, and other steps to prepare the data for training.

(2) Feature Selection: The process of choosing the most informative features to train a model with, which can improve performance and reduce computational cost.

(3) Cross Validation: A method of assessing the quality of a model that involves splitting the data into training and test sets to test its generalization abilities.

(4) Regularization: Techniques such as L1 and L2 regularization are applied to prevent the model from overfitting by adding penalties for the complexity of the model.

(5) Hyperparameter Tuning: Optimizing the parameters of the model that are not trained on the data using methods such as Grid Search or Random Search.

(6) Ensemble Methods: Combining multiple models to improve overall performance. Examples include Bagging (e.g. Random Forest) and Boosting (e.g. AdaBoost, Gradient Boosting).

2.2 Intelligent self-learning programs with functional dynamic combinatorics

Intelligent self-learning programs with functional dynamic combinatorics is an interesting and complex topic that crosses several fields, including machine learning, combinatorics, and dynamic systems. Let’s consider the main aspects:

(1) Self-learning programs: These are systems that can improve their performance based on experience. They use algorithms to analyze data, identify patterns, and adapt to new conditions.

(2) Functional dynamic combinatorics: This method refers to the study of combinations and permutations of objects depending on dynamic changes in the problem conditions. This may involve the use of different combinatorial structures and methods to solve problems that change over time.

(3) Optimization: Self-learning programs can use combinatorial methods to find optimal solutions in complex problems such as routing, scheduling, resource allocation, and others.

(4) Strategies: In problems with changing rules or conditions, such programs can adapt and find new strategies based on previous experience.

(5) Data Analysis: In data analysis tasks, dynamic combinatorics can be used to create models that can adapt to new data and identify new relationships.

(6) Algorithms and Technologies: Various algorithms can be used to implement such programs, including genetic algorithms, decision tree algorithms, neural networks, and other learning methods. Another important aspect is the use of combinatorial optimizations, such as dynamic programming.

(7) Future: With the development of technology and the increase in data volume, self-learning programs using dynamic combinatorics can find new applications in various fields, such as education, economics, medicine, robotics, and others.

2.3 Advantages of intelligent self-learning programs

Intelligent self-learning programs have a number of advantages that make them particularly attractive for use in various fields. Here are some of the main advantages: Intelligent self-

learning programs have a number of advantages that make them particularly attractive for use in various fields. Here are some of the main advantages:

(1) Process Automation: Self-learning programs can automate routine tasks, reducing the need for human intervention and reducing the likelihood of errors, which improves the overall efficiency of processes.

(2) Adaptability: These systems are able to adapt to changes in data over time. They can analyze new data and adjust their models, allowing them to stay relevant without the need for manual updates.

(3) Big Data Processing: Intelligent programs can process and analyze large volumes of data, which allows them to extract useful insights and identify hidden patterns that may not be obvious during manual analysis.

(4) Improving Accuracy: Machine learning systems can continually improve their accuracy as they receive new data. This makes them especially useful in areas where high accuracy is critical, such as medicine, finance, and security.

(5) Personalization: Self-learning programs can use user data to create personalized recommendations and solutions, which increases customer satisfaction and improves the user experience.

(6) Predictive capabilities: Such programs can predict future events based on historical data. This is especially useful in areas such as demand forecasting, financial analysis, and risk assessment.

(7) Innovation and new opportunities: Using intelligent self-learning programs can lead to the development of new products and services, as well as the improvement of existing ones, opening up new business opportunities.

(8) Cost reduction: Automation of processes and increased accuracy can lead to lower labor and resource costs, making the business more efficient.

(9) Improved customer service: Intelligent systems can be used to create chatbots and virtual assistants that provide 24/7 customer support and quickly resolve customer issues.

(10) Anomaly Detection: Self-learning programs can effectively detect anomalies and deviations in data, which is useful for detecting fraud, cyber threats, and other abnormal events.

The fundamental advantage of intelligent self-learning programs is the ability to visually edit your own algorithm. This ability is necessary and sufficient for writing self-learning programs. As a result, intelligent self-learning programs open up new horizons for business, science, and technology, providing more effective and accurate solutions for a wide range of problems.

3 Improving the intelligent self-learning program for finding solutions in the language Author

Intelligent self-learning programs satisfy three conditions: 1) replication, the ability to reproduce and produce digital offspring; 2) variability, the ability to change and differentiate between digital scenarios; 3) improvement of digital offspring and scenarios.

It is possible to configure programs for self-improvement of self-learning using scientific data confirmed by practice [1]. The logic of self-learning is set by the developer in the schemes of program algorithms. The logic of self-learning operates by modifying algorithms and data in the direction of improving solutions.

The Author language interpreter integrates synergetic cognitive queries to subject knowledge ontologies, covering different AI strategies: 1) Directs interdisciplinary cognitive logic of working with algorithms according to the developer's criteria; 2) Determines the knowledge that should be in memory at any given moment; 3) Implements the process of forgettingineffective decisions.

Let's consider a self-learning intelligent program for marking and issuing effective solutions with prime numbers for a variety of areas of activity in the Author programming language. Based on the ability of the "Author" language, changes are made to the native code of programs by creating a repository of marked effective solutions within the program code in the range from 1 to 200 with activity area numbers inside the program code.

```
// prosto.txt
//. self-learning program for finding effective solutions
var isProsto(n){
	numbers of effective solutions m={2,3,5,7};
	if(n<0)return #;
	ok=0;
	if(n==0 || n==1)ok=1;
	for(i=0;i<m.size();++i){
		if(n==m[i]){ok=1;break;}
		if(n<m[i])break;
		}
	if(i==m.size()){
		u=m[i-1];
		oknew=0;
		do{
			++u;
			uok=1;
			for(i=0;i<m.size();++i)if(!(u%m[i])){uok=0;break;}
			if(uok){
				oknew=1;
				m.push(u);
				if(u==n)ok=1;
				}
			}while(n>u);
		if(oknew){
			f=getFunction(getThisFunctionName());
			pos=f.Root();
			pos=f.Next(pos);
			f.setComand(pos,"m="+m.export());
			}}
	return ok;}
void main(){ok=isProsto(n=200); trace("Число "+n+(ok?"":" не")+" effective solution.");}
```

The task is to call the function “isProsto(n)” with a given number for analysis. Actually, the variable “m” contains our storage. To replace the command in the algorithm scheme, the function “f.setComand(pos,“m={solution 2, solution 3}”);” is used, which must be called from the function object. The first parameter must specify the identifier of the node with the command in the graph scheme of the algorithm, which (the command) should be replaced, and the second is the command object (the operator tree). The second parameter can also be a text string that will be implicitly transformed. In order to get the node identifier, we use the fact that the array/storage is on the first node from the beginning of the function algorithm. The function “f.Root()” will return the identifier of the first and last node of the scheme, so to speak, the node of the beginning of the end of the algorithm. From it (the node) you can go to, guaranteed, one, the first node. But moving up from the first and last node (“f.Up(pos)”) it is possible to get a set (an array of identifiers) of nodes, which end the algorithm. The point is that at the end of the algorithm there may be a conditional operator with a branch leading to the beginning of the end node.

After running the program, the function “isProsto(n)” was transformed. The Author language has the ability to use labels, which can be used to find the identifiers of the nodes corresponding to them in the algorithm scheme. The program needed to perform complex calculations that only needed to be done once, so as not to waste time on the same calculations every time the program was launched. The ability of the language to transform the script is used to solve this problem.

```
// one.txt
void main(){
	trace("Helloy World!!!");
	<label:10>
	if(1){
		x=1+1; // сложные вычисления
		f=getFunction(getThisFunctionName());
		pos=f.getLabel(10); // ищем метку
		pos=f.insertDown(pos);
		f.setCommand(pos,"x="+x);
		pos=f.Down(pos);
		command=f.getCommand(pos);
		command.setSub({0},PROGRAM("0"));
		f.setCommand(pos,command);}
	trace(x);
	getstring();}
```

Duplicate code from which the interpreter will take the program code after the first launch:

```
// one.code
void main(){
        trace("Helloy World!!!");
        <label:10>
        x=2;
        if(0){
                x=1+1;
                f=getFunction(getThisFunctionName());
                pos=f.getLabel(10);
                pos=f.insertDown(pos);
                f.setCommand(pos,"x="+x);
                pos=f.Down(pos);
                command=f.getCommand(pos);
                command.setSub({0},PROGRAM("0"));
                f.setCommand(pos,command);
                }
        trace(x);
        getstring();}// one.code
```

The language has a special system function ”Spirit();”, which deletes itself the first time it is executed. It takes the name of the function and arguments to it. The function will be called only once and no traces will remain of it.

```
// Spirit.txt
firstprocess(namef,n){
        x=100*(1+1);
        f=getFunction(namef);
        pos=f.getLabel(n);
        f.setCommand(pos,PROGRAM("k="+x));
        return 1;}
void main(){
        Spirit("firstprocess",getThisFunctionName(),10);
        <label:10>
        trace("k="+k);
        getstring();}
```

The program will display an effective solution and transform into a digital offspring with a new scenario. Self-training of the program in the process of implementing the user’s request according to new scenarios increases its professionalism.

The program with the proposed algorithm schemes and self-learning logic provides safe recommendations and professional knowledge and skills to students in various combinations and an interdisciplinary range of activities based on ethical standards [2, 3].

Combining generative and algorithmic logic in the context of unsupervised artificial intelligence leads to powerful models that can generate new data and apply logical rules to process and analyze it. Combining generative and algorithmic logic can occur in the following ways:

(1) Using generative models to pre-process data: Generative models can be used to create synthetic data that can then be processed using algorithmic methods. This is especially useful in cases where real data is limited or unavailable.

(2) Algorithmic control of the generation process: Generative models can be guided by algorithmic methods that determine how and when to generate new data. This is useful in applications where the context or dynamics of a situation must be taken into account.

(3) Hybrid models: Developing models that use both generative approaches and algorithmic methods. For example, in the field of natural language processing, one can use generative models to generate text and then apply algorithmic approaches to analyze and structure that text.

(4) Code generation: Using generative models to create software code that can then be tested using algorithmic logic to find bugs or optimize.

(5) Data generation: Generative models can create data sets for algorithms that can then be used to improve predictions.

(6) Generate game levels based on gameplay logic, allowing for unique and effective levels that adhere to the game rules.

Generative models such as GANs (Generative Adversarial Networks) and VAEs (Variational Autoencoders) can create new data samples based on a training set. These models learn from structured data and can be used to generate images, text, music, and other forms of content.

Combining generative and algorithmic logic opens up new horizons in the development of

AI systems, allowing them to be more adaptive, creative, and effective in solving complex problems. This direction has great potential for further research and application in various fields such as art, science, economics, and technology.

Artificial intelligence can be taught to work by interacting with it here and now, observing its actions and giving it timely, detailed recommendations, which improves its ability to adapt and learn. Artificial intelligence can continue to self-learn even after a human stops giving feedback, using an AI model created based on the developer's data. This approach opens up new possibilities for creating self-learning and adaptive AI systems.

Artificial intelligence can be taught to work by interacting with it here and now, observing its actions and giving it timely, detailed recommendations, which improves its ability to adapt and learn. Artificial intelligence can continue to self-learn even after a human stops giving feedback, using an AI model created based on the developer's data. This approach opens up new possibilities for creating self-learning and adaptive AI systems.

4 Conclusion

Self-learning artificial intelligence (AI) is playing an increasingly important role in student research, providing them with new tools and opportunities for in-depth study and analysis of data. Here are some key aspects that are affected by the use of self-learning AI in educational and research processes:

(1) Data Analysis: Self-learning AI can process large amounts of data faster and more accurately than a human can manually. This allows learners to conduct deeper research, identify hidden patterns, and draw more informed conclusions.

(2) Personalized Learning: AI systems can adapt to the individual needs of the learner, offering personalized recommender systems that help students focus on areas for improvement, thereby improving learning efficiency.

(3) Modeling and Simulation: AI is used to create complex models and simulations that can be useful in a variety of fields such as physics, biology, economics, and social sciences. This gives learners the opportunity to experiment with different scenarios and explore their implications in a controlled environment.

(4) Natural Language Processing: AI can analyze and interpret large amounts of textual information, which is useful for conducting literature reviews and text mining in the humanities.

(5) Creative Projects: Self-learning AI can be used in creative projects such as generating musical compositions, creating visual art, or writing texts, expanding students' capabilities in the field of art and design.

(6) Automation of Routine Tasks: AI can automate many routine tasks such as data collection and pre-processing, allowing students to focus on more complex aspects of research.

(7) Ethics and Responsibility: Research activities using AI also raise important questions about ethics and responsibility, encouraging students to think critically and discuss the implications of AI in society.

Self-learning artificial intelligence systems in the field of robotic education are a promising direction that can significantly change the approach to learning. Key aspects of using such technologies:

(1) Adaptive Learning: Self-learning AI can adapt learning materials and teaching methods to the individual needs of each student. This allows for the creation of personalized educational programs that take into account the level of knowledge, learning speed, and preferences of students.

(2) Tutoring Robots: AI-powered robots can act as tutors, providing students with assistance in learning various subjects. They can assess students' progress and offer additional exercises or explanations on difficult topics.

(3) Interactive Learning Platforms: Platforms equipped with self-learning algorithms can analyze students' interactions with materials and adapt the content in real time to improve learning outcomes. Robotic platforms such as LEGO Mindstorms or VEX Robotics can use AI to teach coding, engineering, and logical thinking skills.

(4) Assessment and Feedback: Self-learning systems can automate the assessment process, providing students with instant feedback and recommendations on how to improve their skills.

(5) Social Skills Development: AI robots can be used to develop social and communication skills, especially in children with special educational needs. They can engage students in interactions with others.

The use of self-learning AI in robotic education opens up new possibilities for more effective and personalized learning, helping students develop both academic and practical skills (Figure 1).

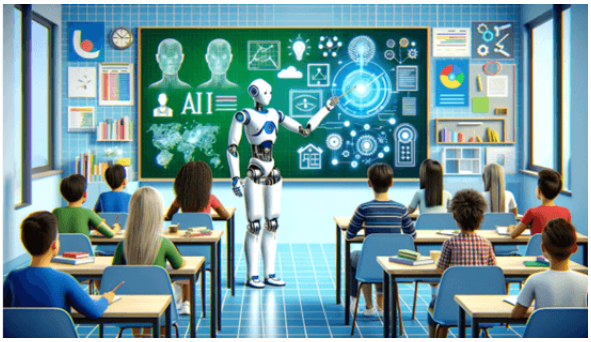

Figure 1 Robotic teacher with self-learning AI

The use of self-learning AI for educational and research purposes opens up new horizons for students, allowing them to develop multi-inter-trans research skills that will be in demand in the future [4–8]. However, it is important to remember the ethical aspects and the need to control the results obtained using AI [9–20]. Based on self-learning AI, it is possible to develop smart wearable technologies as well as intelligent collaborative technologies [21,22].

Conflicts of interest

The author declares that there is no conflict of interest.

References

[1] Bryndin E. Creation of Multi-purpose Intelligent Multimodal Self-Organizing Safe Robotic Ensembles Agents with AGI and cognitive control. COJ Robotics & Artificial Intelligence (COJRA). 2024, 3(5): 1-10.

[2] Bryndin E. Formation of reflexive generative A.I. with ethical measures of use. Research on Intelligent Manufacturing and Assembly. 2024, 3(1): 109-117. https://doi.org/10.25082/rima.2024.01.003

[3] Saxena AK. Society and the Machine: Navigating Ethics in the Age of Artificial Intelligence. 2024, 127. https://www.amazon.com

[4] Furze L. Practical AI Strategies: Engaging with Generative AI in Education. Amba Press. 2024, 198.

[5] Mastrapasqua F. Natural technologies for sustainability. The Project Repository Journal. 2021, 10(1): 128-132. https://doi.org/10.54050/prj10128132

[6] Bowen JA, Watson CE. Teaching with AI: A Practical Guide to a New Era of Human Learning. Johns Hopkins University Press. 2024, 280.

[7] The Future of AI in Education: Personalized Learning and Beyond. 2024. https://www.hp.com/in-en/shop/tech-takes/post/ai-in-education

[8] Oyebola Olusola Ayeni, Nancy Mohd Al Hamad, Onyebuchi Nneamaka Chisom, et al. AI in education: A review of personalized learning and educational technology. GSC Advanced Research and Reviews. 2024, 18(2): 261-271. https://doi.org/10.30574/gscarr.2024.18.2.0062

[9] Makridakis S. The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. Futures. 2017, 90: 46-60. https://doi.org/10.1016/j.futures.2017.03.006

[10] Nyashenko S, Zvereva A, Golubtsova E, et al. The impact of digitalization on the client-oriented approach in the provision of public services to business entities. IOP Conference Series: Materials Science and Engineering. 2020, 940: 012061. https://doi.org/10.1088/1757-899x/940/1/012061

[11] Green E, Singh D, Chia R, et al. AI ethics and higher education: good practice and guidance for educators, learners, and institutions. Published online 2022. https://doi.org/10.58863/20.500.12424/4146302

[12] Kalna F, Belangour A. A Meta-model for Diverse Data Sources in Business Intelligence. American Journal of Embedded Systems and Applications. 2019, 7(1): 1. https://doi.org/10.11648/j.ajesa.20190701.11

[13] Babulak E. Third Millennium Life Saving Smart Cyberspace Driven by AI and Robotics. COJ Robotics & Artificial Intelligence. 2021, 1(4).

[14] Farooq MS, Tahseen R, Omer U. Ethical Guidelines for Artificial Intelligence: A Systematic Literature Review. VFAST Transactions on Software Engineering. 2021, 9(3): 33-47. https://doi.org/10.21015/vtse.v9i3.701

[15] Leonov V A, Kashtanova E V, Lobacheva A S. Ethical aspects of artificial intelligence use in social spheres and management environment. European Proceedings of Social and Behavioural Sciences. 2021, 106: 989-998. https://doi.org/10.15405/epsbs.2021.04.02.118

[16] Hallamaa J, Kalliokoski T. AI Ethics as Applied Ethics. Frontiers in Computer Science. 2022, 4. https://doi.org/10.3389/fcomp.2022.776837

[17] Riczu Zs. Recommendations on the Ethical Aspects of Artificial Intelligence, with an Outlook on the World of Work. Journal of Digital Technologies and Law. 2023, 1(2): 498-519. https://doi.org/10.21202/jdtl.2023.21

[18] Shelton K, Lanier D. The Promises and Perils of AI in Education: Ethics and Equity Have Entered the Chat. Lanier Learning. 2024, 189.

[19] Singh G, Thakur A. AI in Education. The Ethical Frontier of AI and Data Analysis. Published online April 12, 2024: 18-38. https://doi.org/10.4018/979-8-3693-2964-1.ch002

[20] Pan K. Ethics in the Age of AI: Research of the Intersection of Technology, Morality, and Society. Lecture Notes in Education Psychology and Public Media. 2024, 40(1): 260-262. https://doi.org/10.54254/2753-7048/40/20240816

[21] Khristopher Y. (2024).Wearable tech: Fashion meets functionality in 2024. Android Headlines. https://www.androidheadlines.com

[22] Kazerani F. The Pulse of Progress: Smart Wearables and the Evolution of Cardiovascular Monitoring Technologies. International Journal of Research In Science & Engineering. 2024, (44): 30-36. https://doi.org/10.55529/ijrise.44.30.36