[10] Wąs, J., Gudowski, B., Matuszyk, P. J. (2006). Social Distances Model of Pedestrian Dynamics. Cellular Automata, 492–501. doi: https://doi.org/10.1007/11861201_57

[11] Büttner, T., Schwager, R., Stegarescu, D. Agglomeration, Population Size, and the Cost of Providing Public Services: An Empirical Analysis for German States. Discussion Paper No. 04-18. Available at: ftp://ftp.zew.de/pub/zew-docs/dp/dp0418.pdf

[12] Frère, Q., Hammadou, H., Paty, S. (2011). The range of local public services and population size: Is there a "zoo effect" in French jurisdictions? Dans Recherches économiques de Louvain, 77, 87–104. doi: https://doi.org/10.3917/rel.772.0087

[13] Álvarez-Ayuso, I. C., Condeço-Melhorado, A. M., Gutiérrez, J., Zofío, J. L. (2014). Integrating Network Analysis with the Production Function Approach to Study the Spillover Effects of Transport Infrastructure. Regional Studies, 50 (6), 996–1015. doi: https://doi.org/10.1080/00343404.2014.953472

[14] Crainic, T. G., Ricciardi, N., Storchi, G. (2009). Models for Evaluating and Planning City Logistics Systems. Transportation Science, 43 (4), 432–454. doi: https://doi.org/10.1287/trsc.1090.0279

[15] Brilon, W., Hartmann, D. (2004). Fortentwicklung und Bereitstellung eines bundeseinheitlichen Simulationsmodells für Bundesautobahnen. Research project FE01/157/2001/IRB for the Bundesanstalt für Straßenwesen (Federal Highway Research Institute, Germany), in cooperation with the Ruhr-University Bochum.

[16] Bundesministerium für Verkehr, Bau- und Wohnungswesen (BMVBW). (2004). Neubau von Bundesautobahnen.

Received date 19.07.2019
Accepted date 06.11.2019
Published date 30.11.2019

SCRATCH LANGUAGE OF PROGRAMMING VS ENGLISH LANGUAGE: COMPARING MATHEMATICAL AND LINGUISTIC FEATURES

Nataliia Lazebna[1]
natalialazebnaya@gmail.com

Yuliya Fedorova
Department of English Language
Mariupol State University
129a Budivelnykiv ave., Mariupol, Ukraine, 87500
julfedorova84@gmail.com

Mariia Kuznetsova[1]
kuznetsovamariaalexandrovna@gmail.com

[1]*Department of Theory and Practice of Translation*
National University "Zaporizhzhia Polytechnic"
64 Zhukovskogo str., Zaporizhzhia, Ukraine, 69061

Abstract

This paper focuses on Scratch language of programming and traces its math and linguistic features. From a complex consideration about Scratch language programming in linguistic paradigm, focusing on structural, semantic and syntactic features and logic of its narration, this research attempts to clarify specifics of the language and correlate it with the English language features. Global integration of ideas and sciences underline the crucial importance of programming and language conglomerate. Human-computer interfaces, software systems, and development of various programming languages depend on well-balanced structure, shape, logic, and appearance of the actual code. Dynamic characteristics of the Scratch programming environment sustain the creation of interactive and media-rich projects. Ad expansion of Scratch for mediation of animated stories, music videos, science projects, tutorials, and other contents necessitates multifaceted analysis of this programming environment and evokes

the interest of researching Scratch from the math and linguistic perspective as one possible projection on various aspects of the considered programming language.

Keywords: scratch, programming language, mathematics, semantics, syntax, motion blocks, errors, slots, data typing.

DOI: 10.21303/2461-4262.2019.00982

1. Introduction

Currently, global tendencies influence the lives of humans, their activities, and all spheres of their lives. The global integration of different nations results in blending and fusion of various objects and processes [1]. A comprehensive approach to the solution of different problems and issues is of crucial importance. Hypothetically incompatible ideas/objects/agents/processes etc. are the most challenging issues of the modernity. Mathematics and Linguistics, Information Technologies and Humanities complement each other thus enriching their sets of research tools and background thus creating the most favorable basis for the modern breakthroughs.

The aim of this article is focusing on the similarities and differences between the Scratch language of programming and the English language through a prism of mathematics. Syntax and semantics of the former language resemble the main features of the latter. Which of them is a hen, and which is an egg? Which is the most dominant and meaningful one?

The relevance of this paper is a holistic overview of similar and different features characterizing human and programming languages. Focusing on basic command blocks, errors, semantic and structural characteristics of Scratch language of programming, the research attempts to find common and contrasting points between the English language and Scratch language of programming. In previous researches and studies in the field of computer sciences, Scratch language was considered in terms of programming. This paper represents an interdisciplinary approach to the study of this language. Both, alphanumeric basis of Scratch language of programming and words representing basic commands/characteristics of this environment is a perfect opportunity to conduct a study integrating mathematical and linguistic features.

From the perspective of mathematics, Scratch contains arithmetic operations (addition, subtraction, multiplication, division, etc.). The existing set of operations enables tasks performed within the Scratch environment using a linear algorithm with operations on numeric data. In Scratch, it is possible to work out the priority rules for arithmetic operations when calculating mathematical expressions by creating scripts with green blocks nested into each other. The main principle of Scratch work is a request of a natural number from the keyboard giving a message about the parity or oddness of the entered number. Commands from the Sensors blocks are used (a request for initial natural number), Data (to create a variable defining the number itself), Operators (the use of an arithmetic operation). Such commands as: "Determine the parity or oddness of a natural number", "Calculate the area of a rectangle", "Angles. Right and developed angle", "Round of decimal fractions", "Arithmetic average" are performed using arithmetical commands and can be verbalized in the English language. Thus, the Scratch environment creates projects aimed at solving mathematical problems. At the same time, verbalized commands of Scratch language create favorable conditions for analyzing both mathematical and linguistic features.

In the result of this research, it is possible to shed light on a potential generalization of command blocks, errors and other peculiarities of different languages of programming synthesizing them for future practical implementation in the developments of applied linguistics.

In the modern paradigm, the impact on cognitive tool results in the internal information channels of humans or "sensors of consciousness" and complex technical devices. At the end of the 20th century, the role of such tools is important for computers, penetrating into all spheres of human existence. That is a computer can be considered not only as a subject of study but as a tool to help learning and thinking. The "instrumental" concept of L. Vygotsky and J. Dewey focused on objects helping to master the intellectual operations. The opportunities of digital technologies are much broader and they are not limited to methods of educational information visualization. It is well-known that the basis of object-oriented programming is the object, which interacts with each other by sending messages. In response to the received message, the object calls a subroutine

(method). In the Scratch environment, such objects are visual dynamic objects or sprites. With regard to Scratch, it is possible to talk about the so-called "block programming", as the program (scripts) is constructed from multi-colored blocks, where we can think about Lego. Scratch Project is preceded by such developments by M. Reznik as Starlog and NetLogo, which are multi-agent versions of the Logo language (1967) by a group of scientists led by Professor Seymour Papert. Thus, Scratch reflects the educational philosophy of the Logo [8]. The Logo is based on the ideas of cognitive constructivism by Jean Piaget, J. Bruner, constructionism by S. Papert, as well as a number of researchers who criticize instructions, the reductionist approach to education and believe that using a computer as a tool of knowledge a constructive environment is created [9].

Instrumentation connects the quality of education with the degree of mastering instruction, a system of rules that contribute to the organization of the cognitive process as a transfer-receiving data, and patterns of behavior from the translator to the recipient. In this case, as S. Papert underlines, the computer appears as a tool for creating such instructions, which leads to the idea of an automated system. A human is in the center of the Scratch task area, and his interests, opportunities, and actions play a crucial role while dealing with the interface. This software helps individuals not to copy information, but to get it. From the perspective of constructivism, individuals need tools to help them learn, think, solve problems, search and sample data, conduct research, accept mistakes, learn from a personal experience and exchange ideas with others.

It is possible to claim that constructivism, a reliable basis for the Scratch environment, integrates any methods and techniques used and reflects a function of cognition. In this context, constructivism is a conceptual view of the alignment of the educational process responsible for knowledge generation rather than a simple reflection. A childlike curiosity of Scratch language user motivates him to be not a passive recipient, but a constructive personality, a researcher. This type of research ability and competence can be compared with the process of language acquisition. In terms of a constructivist approach, each learner has the right for his/her own opinion, interpretation of certain facts despite possible errors. This approach is applicable only at a certain stage of the learning process both during language acquisition and Scratch language operation.

Nowadays, it is necessary to pay special attention to the potential of digital technologies in the learning process. If it is possible to separate the learning process from reality, then the learners would have an option of using their acquired knowledge or experience for educational purposes. For example, if modern children are interested in computer games where they become active participants of the events and explore the world through its virtual model, then they can implement this strategy for the learning educators implement the concept of "gamification of education". Scratch developers claim that it is possible to prohibit computer games and implement relevant tools to create one's own learning environment. In this case, the computer is not a subject for the study, but only a tool for creating a project with the main focus on an individual.

2. Methods of research

This paper implements a literature review to analyze theoretical underpinnings of the considered theme, correlation of Scratch language of programming, mathematical and English language features, and analysis of figures and diagrams of Scratch programming language in terms of logic, coherence and lingual concerns.

Focusing on the basic command blocks of Scratch language, their dynamic basis is compared with the verbal representation of human actions by means of verbs. Enlivened nature of Scratch language of programming resembles human language features. Further analysis of static and dynamic data typing is another key concern of this study. Based on the abovementioned processes, these two types of typing are correlated with data used in Scratch language of programming.

3. General discussion and results

This study considers both mathematical and linguistic features of Scratch language of programming. For the first time, this paper attempts to integrate both alphanumeric mathematical basis of Scratch and verbal representation of programmed commands. This paper is the first attempt to draw further conclusions about the common basis between language and programming. In the

given study, enlivened and animated nature of Scratch resembles the main features of a human language "living creature". Both, the speakers and programmers are able to change commands/modes of language, practice editing and fine-tuning to reach a harmony of ideas and their embodiment either in speech/written form (human language) or command editing/improvement in Scratch [1].

Both in the environments of the Family of Logo, driving a bug, and in Scratch, students "teach" (or program the actions) the algorithm performers respond to them, interact with each other, control their position, movement, appearance, etc. At the same time, in the form of a game, they learn the important algorithmic constructions and mathematical concepts. If to focus on the example of the Sierpinski triangle, it is evident that the structural representation of command blocks in Scratch reflects structural features of Scratch language of programming (**Fig. 1, 2**). Moreover, a systematic approach to programming environment creation depends on the implementation of various mathematical tools and methods. Such concepts as integers and fractional, random numbers; plane coordinates; relative dimensions and relative distance; angles and their meaning; polygons, regular polygons; symmetry; circle and its radius, concentric circles, or algorithms. Scratch environment represents a wide range of algorithmic constructions which serve the rules for developing software code. Linear algorithms contain various branches and cycles; variables, their values, and different parameters.

For example, many Scratch projects involve moving sprites around the scene in a given point, a turn at an angle, while students master the concept of angle, a negative number, random number, and work with the coordinate plane. It is possible to use Scratch library and write scripts containing commands, which will make the sprite move along the axes, while providing the coordinates of its location, and the user can set the keyboard coordinates of the point to which the sprite will go. To create an animation effect, it is possible to use the sprite costume change.

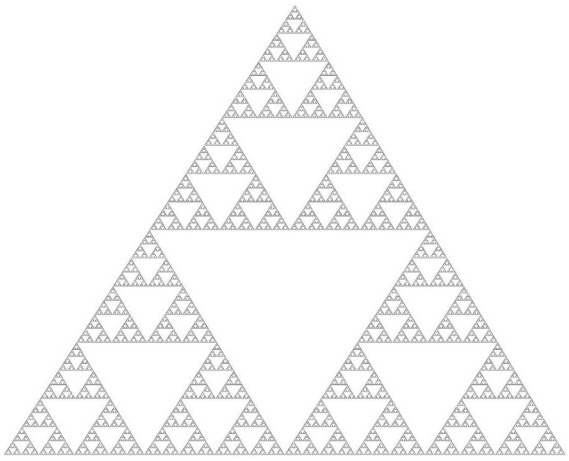

Fig. 1. Sierpinski triangle [14]

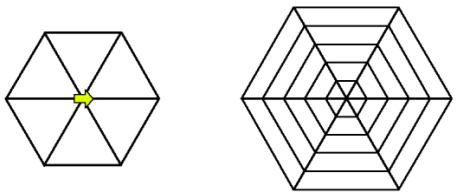

Fig. 2. Scratch procedures with triangles [13]

Time delay or time gaps between changing the sprite's costumes (or modifying his actions) can be regulated by different commands "swim (...) seconds to the point *x* (...), *y* (...)". The users set the values for the coordinates randomly using the "give a random number from ... to ..." [6].

A learner can imagine itself in the role of a sprite and describe his actions within a circle. Created projects can be of different nature: interactive multimedia presentations, demos, simula-

tors, animated stories, etc. As a rule, in such projects, it is necessary to organize a dialogue between sprites through signaling messages and to make up a dialogue between the user and the program based on commands of the "control" and "sensors" units. At the same time, it is possible to achieve both the sequential performances of scripts and parallel actions of many performers. A student as a project author can choose a sprite from the Scratch library, download it from file or web site, or come up with their objects and draw an environment for them. For this, Scratch has a built-in graphics editor. In addition, projects can be voiced using special commands from the sound block. One can also select, record and process some audio fragments. Therefore, to consider the Scratch environment only as a programming system is unfair. Ability to work with various types of media information (text, graphics, sound, animation) makes Scratch multimedia system and creates a didactic basis for the initial acquaintance of students with digital technology, a media-life organization tool, the emergence of the "media culture" to adopt an individual to the conditions of dynamic information society. Moreover, there is a connection between the constructivist educational paradigm and the sociocultural trend, which underlines both communicative and cultural function of the Scratch language of programming and the media used.

Different motivating perspectives about programming language are available. For example, "An understanding of the cognitive underpinnings of computer programming as a human skill would have much to offer in augmenting our fundamental knowledge of general problem-solving. Such awareness would also be of benefit in designing human-computer interfaces, programming languages and other software systems that are user-friendlier" [3].

Currently, programming creates a creative environment, where the syntax represents different programming constructs, such as "structure, shape, logic, and the appearance of the actual code without explicitly understanding other important issues such as efficiency, meaning, purpose, and proper usage of such code" [6].

"Scratch is a visual programming environment that lets users create interactive, media-rich projects. People have created a wide range of projects with Scratch, including animated stories, games, online news shows, book reports, greeting cards, music videos, science projects, tutorials, simulations, and sensor-driven art and music projects" [6].

Scratch commands identify the digital narrative of Sprite's events programmed. Exact instructions, numbers and variables

setting emphasize a non-ambiguous nature of Scratch. In other words, both, the programmer creates/writes/sets exact commands. Their consequential completion by a user results in an animation of the Sprite. The environment, surrounding or any other external factors do not change the process of commands completion. This "tinkerability" supports an improved approach to writing scripts. A programmer uses a block by ticking on it, changing or modifying it. Therefore, a key concern is to focus on the functionality of blocks (**Fig. 3**).

There are the following blocks in Scratch:

1) motion;
2) looks;
3) sound;
4) pen;
5) control;
6) sensing;
7) numbers;
8) variables.

This set of commands represents both parameters validation and general functions performance (**Fig. 3**). The Sprite named Marty follows these commands, but if there are too many of them, "its feet will hit each other and bad things will happen. To avoid accidentally sending a bad command, it is possible to 'bound' the turn amount to within certain limits" [6]. There can be certain clashes between parameters and functions. Therefore, an artificially created programmed character can stop functioning if there are too many commands given.

It is possible to draw parallels between the external factors preventing an adequate perception of the initial message. Pronounced words, coined terms or narrated sentences look clear and

well-structured and comprehensible, though the external conditions and social environment shift the accents of linguistic tools. For example,

"He's ice cold." [a reference to a dead body]

"He's ice cold." [a reference to a child sleeping during a cold night]

"He's ice cold." [a reference to a silent eye-witness during prosecution]

"He's ice cold." [a reference to a lack of emotions.]

Different contexts of perception of the same phrase, which is grammatically and lexically identical, diversify its perception.

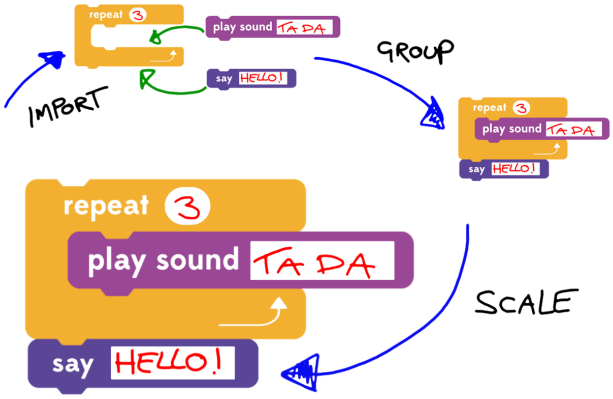

Fig. 3. Scratch parameters validation and general functions performance [15]

The concept of a mistake is incompatible with Scratch. Unlike the dynamic and fluid nature of phonemes/morphemes/words and other "bricks" of human language, blocks of Scratch are matched to make sense. According to Maloney et al (2010), "Rather than failing with an error message, every block attempt to do something sensible even when presented with out-of-range inputs. For example, the 'set size' block bounds the range of its parameter so that it can't make the sprite excessively large (possibly exceeding system limits) or invisibly small" [5]. For a user, it is of crucial importance to avoid mistakes. Error-free scripts writing should follow the trouble-shooting strategy.

However, in Scratch, unlike a text-based language, where the context is dominant, there is a high cost to improve the command set. There are 92 command blocks in Scratch 1.0. The number of commands should be controlled. Additional commands lead to obsolete blocks. If to implement the scientific math function block and the image effect blocks, it is relevant to appeal for individual blocks value.

If to project this idea on a text-based language, redundant words or they're inappropriate usage can lead to a misperception of the initial text message [2]. For example, redundant words in the following sentences result in pleonasm, which is "the use of extraneous words in an expression such that removing them would not significantly alter the meaning of the expression" [10].

Freshly squeezed and no additives, just plain pure fruit pulp.

The dressing is absolutely incredibly fabulously flavorful.

The first sentence contains one redundant word (plain), and the second sentence includes other two redundant words (incredibly fabulously).

Anyway, the occurrence of mistakes is possible both in Scratch and human language. The role of syntax is dominant and combinatory principles are of crucial importance in these two types of languages. The following table illustrates a consequential completion of commands and their similarity with the principles of a puzzle making or LEGO construction (**Fig. 4**).

According to one of the recent studies, "some students continue to 'think in Scratch blocks' as a form of pseudo-code, even after moving to the text-based language" [4]. This mode of thinking resembles a common feature of human language speakers, who use verbs as exact statements and joints/prepositions/particles as operators.

"Command blocks are like the statements of a text-based language; function blocks are like operators. Function blocks are not joined in linear sequences like command blocks. Instead, they are used as arguments to commands and nested together to build expressions" [7]. There are also trigger blocks, which connect event (e. g. mouse clicks and key presses). The digital and symbolic nature of Scratch is expressed in three first-class data types: Boolean, number, and string [4]. A visual exponent of data is represented in a function block.

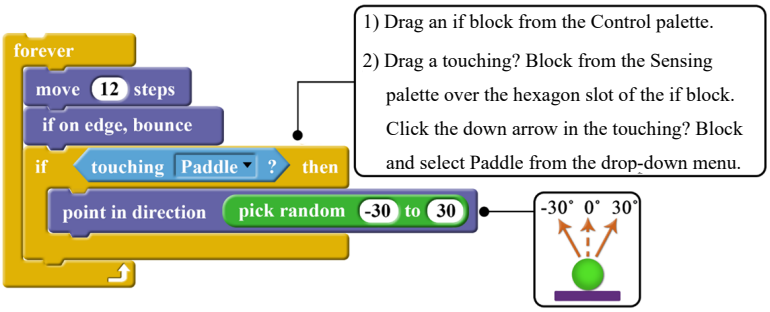

Fig. 4. Scratch block types with codes [12]

Set and stable Boolean parameter slots contrast with the less strict number and string parameter slots. The context defines the relationship between numbers and strings. Therefore, it is relevant to extend visual grammar and handle additional first-class types in the future.

A common thread is a choice between static and dynamic typing in text-based languages. "The advantages of static typing include earlier detection of programming mistakes (e. g. preventing adding an integer to a Boolean), better documentation in the form of type signatures (e. g. incorporating number and types of arguments when resolving names), more opportunities for compiler optimizations (e.g. replacing virtual calls by direct calls when the exact type of the receiver is known statically), increased runtime efficiency (e. g. not all values need to carry a dynamic type), and a better design time developer experience (e. g. knowing the type of the receiver, the IDE can present a drop-down menu of all applicable members)" [7].

According to supporters of dynamically typed languages, "static typing is too rigid, and the softness of dynamical languages makes them ideally suited for prototyping systems with changing or unknown requirements, or that interact with other systems that change unpredictably (data and application integration)" [7]. Static data can lead to some errors. Meijer claims, "Static typing is a powerful tool to help programmers express their assumptions about the problem they are trying to solve and allows them to write more concise and correct code. Dealing with uncertain assumptions, dynamism and (unexpected) change is becoming increasingly important in a loosely distributed world instead of hammering on the differences between dynamically and statically typed languages, we should instead strive for peaceful integration of static and dynamic aspect in the same language. Static typing where possible, dynamic typing when needed!" [7]. Further representation of differences between static and dynamic typing is shown in **Fig. 5, 6.**

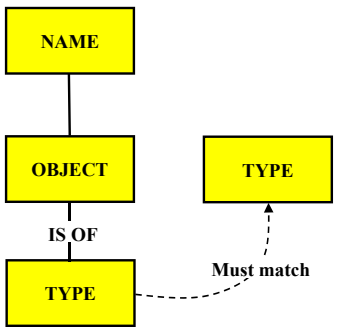

Fig. 5. Static typing [7]

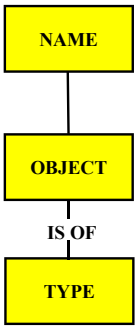

Fig. 6. Dynamic typing [7]

Thus, the problem of data compilation is of crucial importance for the users of Scratch. The assembling of program blocks is a great contribution to the concurrency model of the program. Animated nature of Scratch environment underlines the enlivened nature of the system "with no run/edit switch, so commands or code snippets can be run with a click, and graphical feedback shows execution. Variables and lists have concrete visualizations, so the effect of data operations can be seen immediately" [7]. Therefore, an operator can focus on his actions and their immediate reflection on the screen.

4. Conclusions

This study is the first and unique attempt of correlation between mathematical and linguistic features of Scratch language. Going beyond the scope of programming, this paper shows an interdisciplinary nature of the considered language. The application of alphanumeric features of Scratch language of programming, which are further verbalized in the English language, underline a holistic integration of mathematics and linguistics as two main identifiers of the language nature. Static and dynamic options of Scratch data typing, symbolic and digital nature of this programming language (environment) and other abovementioned features remind of human language. Further correlation of programming languages semantics, syntax and pragmatics is a favorable background for research in this field. Thus, it is relevant to appeal for the cognitive linguistics mechanisms in studying modes of perception and verbal representation or commands performance of Scratch language users. An interdisciplinary approach to different languages of programming and their analysis highlights further correlations between mathematics, linguistics, computer sciences, computer lexicography, and other disciplines. Block thinking, semiotics and universal essence of modern communication develop a creative outlet for post-modern individualized integration of knowledge and its processing by the contemporaries despite language chosen: from any language of programming to any human language.

Moreover, it is relevant to highlight detailed comparing/contrasting of both human and programming languages. Due to a generalized nature of the given research focusing on Scratch language of programming, it is of vital importance to find similarities and differences in various types of programming languages to foster developments in the field of applied linguistics, such as compilation of parallel corpora data, automated machine translation principles correction, grammar checking programs and many other related applications. This paper is the first attempt on the background of correlated human and programming languages, which enables implementation of found principles in the field of applied linguistics, humanities and mathematics and informatics. Especially, findings of the integrative nature of Scratch language are appropriate for further consideration in the field of applied linguistics, a perspective, and innovative science. There is a lack of studies focusing on the correlation between linguistic and programming. To bridge the gap, one should concentrate on mutual cooperation between programmers and linguists. In this case, it will be easier to avoid semantic mistakes in automatic translation, find mechanisms of facilitating the management of parallel texts corpora, management of grammar checkers and other related programs. Frequency word books also deserve special attention of these specialists. Firstly, the manual creation of such dictionaries is a long-term and challenging process. For

specialists in this area, it is relevant to correlate thousands of texts with millions of words. If to apply for a program on Python, correlation of millions of words can be mediated within several seconds. It is important to delve into texts and identify patterns in them, including mathematical or statistical ones. If specialists determine the gender or age of texts authors, it can be compared with fingerprints. This is a task of neural networks and machine learning. These steps can be mediated by a further detailed study of Scratch, Python and other programming languages to see the projection of enlivened nature of these artificial languages and their potential of correlation with natural languages.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Gabbrielli, M., Martini, S. (2010). Programming Languages: Principles and Paradigms. Undergraduate Topics in Computer Science. Springer. doi: https://doi.org/10.1007/978-1-84882-914-5

[2] Kashefi, O., Lucas, A. T., Hwa, R. (2018). Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). doi: https://doi.org/10.18653/v1/n18-2036

[3] Lenarcic, J. (2004). Behavioural Issues in Software Development: The Evolution of a New Course Dealing with the Psychology of Computer Programming. Issues in Informing Science and Information Technology, 1, 0247–0252. doi: https://doi.org/10.28945/735

[4] Malan, D. J., Leitner, H. H. (2007). Scratch for budding computer scientists. ACM SIGCSE Bulletin, 39 (1), 223–227. doi: https://doi.org/10.1145/1227504.1227388

[5] Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. (2010). The Scratch Programming Language and Environment. ACM Transactions on Computing Education, 10 (4), 1–15. doi: https://doi.org/10.1145/1868358.1868363

[6] May, J., Dhillon, G. (2009). Interpreting beyond Syntactics: A Semiotic Learning Model for Computer Programming Languages. Journal of Information Systems Education, 20 (4), 431–438.

[7] Meijer, E., Drayton, P. (2004). Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War between Programming Languages. Intended for Submission to the Revival of Dynamic Languages. Available at: https://www.ics.uci.edu/~lopes/teaching/inf212W12/readings/rdl04meijer.pdf

[8] Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books, 242.

[9] Piaget, J., Piercy, M., Berlyne, D. E. (1950). The Psychology of Intelligence. London: Routledge & Kegan Paul Ltd.

[10] Pleonasm. Available at: https://www.merriam-webster.com/dictionary/pleonasm

[11] Scratch Blocks. Available at: http://www.mblock.cc/scratch-blocks/

[12] Scratch Marty. Available at: https://www.generationrobots.com/media/ChallengeOutlinesFR.pdf

[13] Shapes and Fractals. Available at: https://nelson.coderdojo.nz/projects/shapes-and-fractals/

[14] Sierpinski Triangle. Available at: http://jwilson.coe.uga.edu/emat6680/parsons/mvp6690/essay1/sierpinski.html

[15] Scratch Blocks. Available at: http://scratched.gse.harvard.edu/resources/vector-scratch-blocks.html

Received date 27.05.2019
Accepted date 19.08.2019
Published date 30.11.2019