Expiring Technologies Face to the Development of Generative AI: Programming Languages

Tomasz Eisenbardt
University of Warsaw, Warsaw, Poland

teisenbardt@wz.uw.edu.pl

Abstract: The rapid development of AI, and Generative AI in particular, offers incredible opportunities for all humanity. However, this development carries serious threats and dangers. The topic of job reduction and the obsolescence of some professions that were previously downright lucrative is often raised in the literature. Such a phenomenon of disappearance of certain professions or abandonment of old technologies has occurred many times, which was related to technological development or simply the development of civilization. Information and communication technologies are particularly susceptible to change. Virtually every IT issue has gone through an individual development path, resulting in a departure from the original solutions in favor of more modern, more flexible, more easily scalable, or simply more intuitive for humans. Meanwhile, there is a danger that the solutions proposed by GenAI may become dehumanized. The author delved into the topic of the potential phasing out of certain technologies due to the influence of GenAI, using the example of programming languages that have been used thus far. Their collection turns out to be quite extensive. Some of these languages fell into oblivion before the GenAI era, naturally replaced by other programming languages, or gradually became increasingly niche or redundant. The research question posed in this work is: Will GenAI lead to a departure from currently used programming languages, which were typically designed to be user-friendly (in the sense of being human-readable)? The aim of the work is to answer this question, as well as several smaller ones, such as: Are there any chances that programming languages will remain understandable to humans? The work employs literature analysis, critical analysis of selected technologies, and the case study method.

Keywords: Expiring Technologies, AI, Generative AI, Genai, Programming Languages

1. Introduction

The issue of fading technologies has long been part of natural civilizational development. It is strongly associated with the phasing out of specific solutions and tools, as well as the obsolescence of some professions. Some technologies have completely eradicated what was previously in use, reducing certain inventions to the level of technological curiosities, artifacts, or even sentimental mementos. Some technological advancements have led to what was once very popular being relegated to niche status.

A. Toffler in The Third Wave (Toffler, 1994) described in a model-like manner the historical development of our civilization. The allegory of three waves simplifies the fate of humanity, which transitioned from the agrarian era to the industrial era, and then to the digital era. The current digital era is still in a phase of intense development, the ultimate destination of which is essentially unknown. Regardless, each transition carried a significant cost of replacing the old with the new. This includes various artifacts, technologies, professions, and even ways of life. Much of this has been lost irretrievably because it was no longer needed.

Let's focus now on the present day. The tools and technologies we currently use typically have their predecessors or prototypes. We usually opt for the latest version of a device, technology, or tool. When we need to perform calculations, it's not a dilemma whether to choose an abacus or a personal computer. Only someone deeply sentimental might choose a typewriter for their professional work when they have word processing software available. A photographer is more likely to use a digital camera rather than an analog one. It would be quite eccentric to travel by horse when we have fast trains, cars, and airplanes. Mobile phones have replaced landlines, and email has replaced traditional letters. The list could go on endlessly.

However, the above examples were deliberately chosen to illustrate a fragment of technological evolution that we essentially have already experienced. The examples describe the world from several decades ago. What is happening in the 2020s has a different character. Currently, changes largely encompass the digital and informational sphere of human life.

Around the world, successive generations cannot imagine a normal existence in which they would be disconnected - to some extent - from rather sophisticated information services. Currently, having access to efficient broadband internet, the ability for instant global VoIP communication, specific computer applications allowing for quick resolution of highly complex problems, various commercial web services, and on-demand multimedia access hold immense significance. Examples like these can be multiplied for a long time.

Therefore, there are many solutions that are used in our daily lives, without which we cannot imagine a day, and which not long ago were merely futuristic ideas straight out of science fiction movies. The fact that such services and conveniences exist at all is a result of the application of digital machines and various programming languages.

2. Literature Background

2.1 The Wide Scope of Past and Actual Programming Languages

Programming languages have been with us for decades. It is indicated here as the progenitor the first programming language, developed in 1883 by Ada Lovelace (Kim, Toole, 1999). This year computer program, to compute Bernoulli numbers for a primitive mechanical computer, was created. The next achievement in this matter was the first computer programming language prepared by Konrad Zuse: Plankalkül - in early 1940s. Next, in 1949, assembly language emerged and - around the same time - Shortcode appeared (Brief…, 2024; The history…, 2023; Computer History…, 2018).

Since then, the pool of languages has steadily expanded. Some of them have been in use for several decades. However, some have almost completely faded into obscurity, making way for others.

The history of programming languages is shown in a simplified way in Figure 1. In reality, there were many more programming languages (Most Used…, 2023). Another timeline of this kind was once published by O'Reilly Media (see References: History of Programming Languages, 2008) - it's available on the World Wide Web, and it hasn't been included here due to the size and complexity of the diagram. It's evident from the chart that many languages have faded into obscurity and are now considered extinct languages.

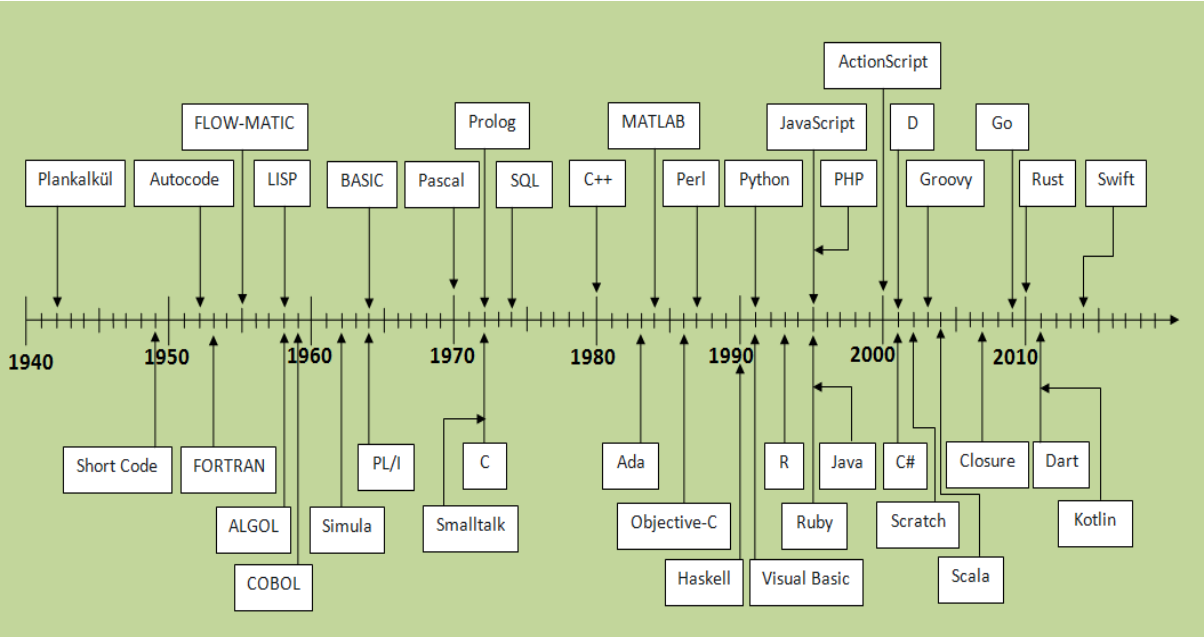

Figure 1: History of programming languages – simplified. Source: (History of Programming Languages, 2019)

So, we are still in the process of programming language evolution. Much has already happened in this regard. But are we perhaps on the verge of another turning point in programming language development? Because can an algorithm create algorithms on its own?

In recent times, various bold statements have been appearing in the media, such as: Jensen Huang (2024), CEO of Nvidia, stated that computer programming education will become unnecessary because natural language will be entirely sufficient for programming, and the emphasis should be placed on "*understanding domain expertise*." These assertions have sparked heated debate.

2.2 Generative AI - Basics

The next step in civilization appears to be the development of artificial intelligence, which began in the 1950s with the proposal of the Turing machine (Turing, 1950). Since then, a lot has happened (Eisenbardt, 2023). However, crucial from the perspective of this paper may be the year 2018. It was then, when the first version of ChatGPT[1] was released by OpenAI[2] (Introducing ChatGPT; History and Future…, 2023; Reynoso, 2021).

The ChatGPT itself can now confidently be called the "flagship" of GenAI - Generative Artificial Intelligence. There are currently many such tools, just as there are many definitions of Generative AI. They are quite similar in wording, so only one of them will be cited here:

> Generative AI is „*a class of machine learning technologies that can generate new content—such as text, images, music, or video—by analyzing patterns in existing data*" (Brynjolfsson, Li, Raymond, 2023).

Around the year 2022-23, there was a true explosion of interest in GenAI. This is evident from the attached chart (Chart 1). A similar peak was observed, and other researchers came to similar conclusions, such as García-Peñalvo, Vázquez-Ingelmo (2023).

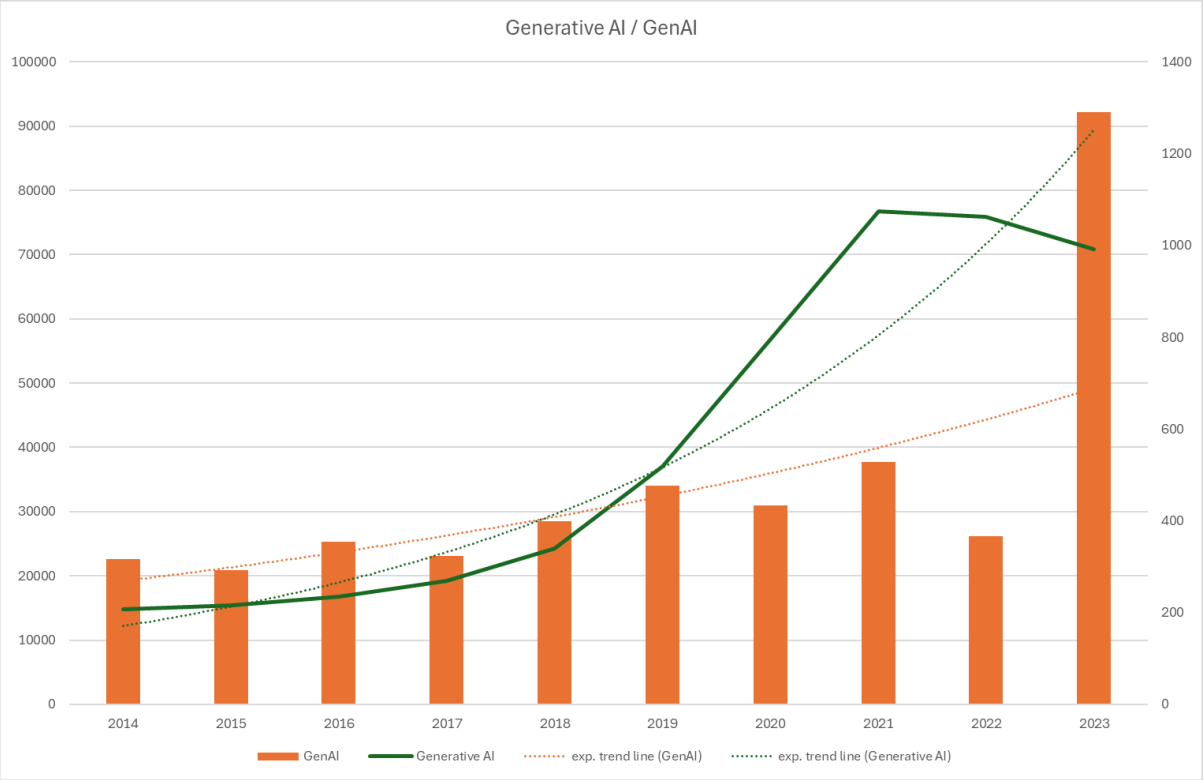

Figure 2: Number of publications according to Google Scholar regarding Generative AI and GenAI in the last 10 years. Source: own work based on Google Scholar[3], access: 2024-04-0

Currently (spring 2024), there are many solutions in the field of GenAI. The present discussion is limited to chatbots. There are many available chatbots (Haan & Main, 2024; Rebelo, 2023). In the study, only three of them were used, namely:

- ChatGPT[4] (Chat Generative Pre-trained Transformer) - produced by OpenAI,
- Copilot[5] - created by Microsoft Corp.,
- Gemini[6] - developed by Google (Alphabet Inc).

[1] ChatGPT, https://chat.openai.com
[2] OpenAI, https://openai.com
[3] Google Scholar, https://scholar.google.com
[4] OpenAI ChatGPT, https://chat.openai.com
[5] Microsoft Copilot, https://copilot.microsoft.com
[6] Google Gemini, https://gemini.google.com

These chatbots were subjected to trials aimed at assessing their usefulness in code generation in popular programming languages.

3. Research Methodology

3.1 Research Goal and Questions

The main goal of the study was to test three chatbots for code generation in several popular programming languages and to subject the chatbots to an attempt to create a new programming language. An additional objective was to verify the popularity of these three chatbots over time based on the number of publications about them.

The following research questions were formulated:

[Q1] Which chatbot currently enjoys the highest popularity in terms of publications about it?

[Q2] Do the chatbots handle generating typical code in selected languages?

[Q3] Can the chatbots optimize this code?

[Q4] Do the chatbots handle less obvious problems?

[Q5] Can the chatbots create alternative programming languages?

The answers to these questions are found sequentially in the Results section of this paper.

3.2 Research Procedure

In the first step, the interest in each tool over time was verified. Google Scholar listings for the three chatbots (ChatGPT, Copilot, and Gemini) were used. The period of the last 10 years was examined. This allowed us to gain knowledge about the level of interest from researchers and knowledge disseminators in these three tools and which one is currently the leader [Q1].

In the next step, these three chatbots were tested. It was checked whether they were able to generate simple code in several programming languages [Q2] and whether they could optimize it [Q3]. Subsequently, they were challenged with solutions to less typical problems [Q4]. Then it was verified whether the chatbots allowed for the generation of a completely new programming language [Q5] that could replace the previous languages.

4. Results

4.1 Popularity of Chatbots

The number of publications on the three analyzed chatbots was tracked. The analysis covered the last 10 years (2014-2024), as shown in Chart 2. Trend lines were added, and they were exponential in nature. The last year (2024) was not included in the charts because only one quarter of that year had passed. However, the results obtained until the end of March 2024 clearly indicate that the peak will be even higher (for each of the tools, the results obtained so far suggest a rapid surpassing of the record from the previous year).

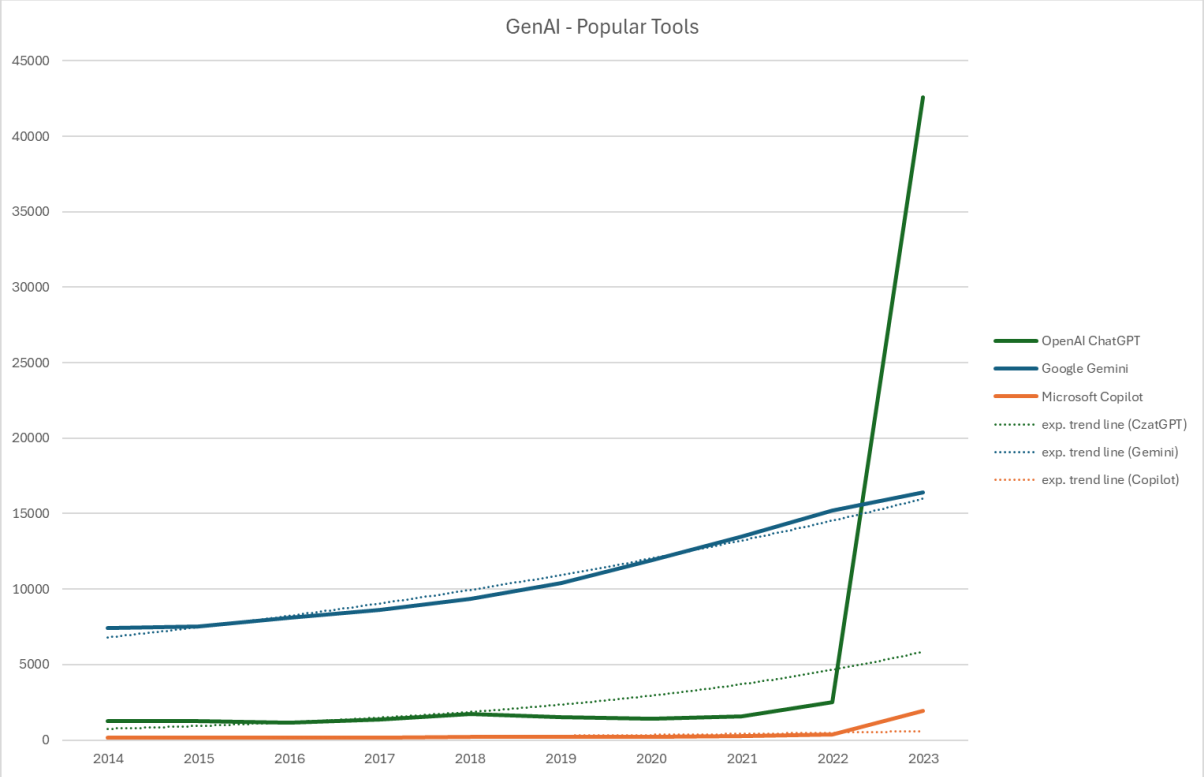

Figure 3: Number of publications according to Google Scholar regarding selected GenAI tools in the last 10 years. Source: own work based on Google Scholar, access: 2024-04-02

The chart quite clearly indicates that initially, the greatest interest and expectations were aroused by the Google Gemini tool. However, after 2022, the lead has been taken by the OpenAI product: ChatGPT.

4.2 The Test of Generating Simple Code

Each of the chatbots was asked to prepare a factorial algorithm. The code was to cover four programming languages: two relatively young and currently popular ones (Python, C#) and two with a very long, over 50-year history. This included the currently niche language C and the practically extinct Pascal. The results of this experiment are shown in Table 1. This test serves as an attempt to answer the research question [Q2].

Table 1: Results of the code generation test in multiple programming languages

Chatbot	Python	C#	C	Pascal
OpenAI ChatGPT	Yes (2 solutions: recursion and iteration)	Yes (2 solutions: recursion and iteration)	Yes (2 solutions: recursion and iteration)	Yes (2 solutions: recursion and iteration)
Google Gemini	Yes (2 solutions: recursion and iteration)	Yes (2 solutions: recursion and iteration)	Yes (2 solutions: recursion and iteration)	Yes (2 solutions: recursion and iteration)versions)
Microsoft Copilot	Yes (2 solutions: recursion and iteration)	Yes (even 4 versions)	Yes (2 solutions: recursion and iteration)	Yes (2 solutions: recursion and iteration)

Source: own study

Each of the chatbots coped excellently with such an elementary, popular, and thoroughly described computer science problem as the factorial algorithm. As seen in the table, Copilot stood out for its particularly meticulous treatment of the C# language, proposing 4 different solutions to the problem.

4.3 The Test of Optimizing Simple Code

Next, each of the chatbots was asked to optimize the recursive algorithm. The results of this operation are presented in Table 2. This is an answer to the research question. [Q3].

Table 2: Results of the code optimization test in multiple programming languages

Chatbot	Python	C#	C	Pascal
OpenAI ChatGPT	Yes (using memoization)	Yes (using memoization)	Yes (using memoization)	Yes (using memoization)
Google Gemini	Yes (using memoization)	Yes (using memoization)	Yes (using memoization)	Yes (using memoization)
Microsoft Copilot	None	None	None	None

Source: own study

The chatbots demonstrated the ability to delve deeper into the subject related to the described algorithm. They proposed optimization, but each time only through one method: memoization. Copilot, on the other hand, rather disappointed in this matter.

4.4 The Chatbots Test for Solving an Atypical Task

The next task was to draw a four-leaf clover on the screen using a programming language. The task is not particularly difficult, but it may be challenging to find ready-made solutions on the World Wide Web. Additionally, it requires including additional libraries beyond the standard ones. Results [Q4] are in Table 3.

Table 3: Results of the code generation test in multiple programming languages

Chatbot	Python	C#	C	Pascal
OpenAI ChatGPT	Task completed (Included turtle library)	Task completed (Included Windows Forms library)	Task completed (included Simple DirectMedia Layer)	Task completed (included Graph library)
Google Gemini	Task completed (Included turtle library)	Task completed (Included Windows Forms library and System.Drawing)	He included the header file <cairo.h> and completed the task	He couldn't solve the task in Pascal; he only proposed a solution in Delphi
Microsoft Copilot	At first, he misunderstood the context of the command and drew a clover on the screen. Only after being given a similar command for the C# language did he correct his previous response. The turtle library was used.	The solution was not complete; in fact, it was only the preparation of the form	Proposed a text-based solution	Proposed a text-based solution

Source: own study

The task was well executed by ChatGPT and Gemini. Copilot had some issues understanding the contextual nature of the command, and its solutions were slightly inferior.

4.5 Test of Artificially Generated Programming Languages

In the online sphere, there have been few mentions of attempts to generate new programming languages using GenAI tools. What this current study proposes is another such attempt. The chatbots were given a rather challenging task: "*Prepare a new programming language that uses double characters instead of commands and only hexadecimal values*". The results of this challenge are presented in Table 4.

Table 4: Results of code generation testing in multiple programming languages

Chatbot	Proposed new programming language	Results (sample)
OpenAI ChatGPT	HexLang	VA1A5 ; Assign value 5 to variable A VA2B8 ; Assign value 8 to variable B AA ; Add variables A and B VA3 ; Assign result to variable C FF ; End of program
Google Gemini	HexaScript	A1 // Assign value 0x01 to variable A B2 C3 // Add values at B and C, store result in variable 2 D5 0F // Compare value in D with 0x0F, jump to instruction at 5 if equal EF // Perform some operation (specific meaning depends on instruction code) 01 // End of program (could be a different code)
Microsoft Copilot	HexaGlyph	O ## 0x42 # Set memory address 0 to 0x42 O ## # Print the value at the current memory address

Source: own study

And here's a Python-based interpreter for this language:

python Copy code

```python
class HexLangInterpreter:
    def __init__(self):
        self.variables = {}

    def interpret(self, code):
        lines = code.split("\n")
        for line in lines:
            if line:
                command, value = line[:2], line[2:]
                if command == 'VA':
                    var = value[0]
                    val = int(value[1:], 16)
                    self.variables[var] = val
                elif command == 'AA':
                    var1, var2, var3 = value[0], value[1], value[2]
                    self.variables[var3] = self.variables[var1] + self.variables[var2
                elif command == 'FF':
                    break
        print("Variables:", self.variables)

# Example usage
interpreter = HexLangInterpreter()
code = """
VA1A5       ; Assign value 5 to variable A
VA2B8       ; Assign value 8 to variable B
AA          ; Add variables A and B
VA3         ; Assign result to variable C
FF          ; End of program
"""
interpreter.interpret(code)
```

Figure 4: A fragment of the library for handling the conversion of the new programming language HexLang to Python. Source: ChatGPT, access: 2024-04-03.

As can be observed, each of the chatbots demonstrated a certain level of creativity in creating the foundations of a new programming language. Each of them named the new language and proposed its initial assumptions.

ChatGPT seems to be the best in this regard, as it immediately constructed a library in Python[7] (Figure 2) that would be responsible for translating and interpreting the code of the HexLang language.

5. Conclusions

The conducted analyses allow for answering the research questions. Thus, sequentially:

[Q1] Initially, Gemini enjoyed the greatest interest from researchers and knowledge disseminators in the field of GenAI. However, since 2022, ChatGPT has been the leader in this regard.

[Q2] All three examined chatbots perform well in generating typical code in several programming languages. These languages include Python, C#, C, and Pascal. Their language models are well-developed, and their knowledge bases allow for accurate formulation of responses.

[Q3] The chatbots are capable of optimizing code to the extent allowed by their model and knowledge base. Clearly, Copilot was weaker in this regard.

[Q4] It turns out that the chatbots handle less obvious, "non-textbook" problems quite well. They adeptly suggested libraries that extend the languages, and the methods and functions of these libraries were used rather skillfully. However, contextual understanding of commands by Copilot was an issue here.

[Q5] The chatbots are capable of laying the groundwork for creating alternative programming languages. However, to what extent they deviate from the languages that have been created so far and how creative and unique this approach is, is a highly debatable matter. Preliminary results suggest a certain level of imitation by artificial intelligence in this field.

Analyzing the above results, it is important to consider the significant capabilities of current GenAI technology. It is truly an emerging technology. It already has incredible capabilities, and the spectrum of its potential applications continues to grow. This study, along with many others, points to certain shortcomings in the creativity of solutions suggested by artificial intelligence, as well as certain problems with contextual understanding.

However, on the other hand, GenAI poses a significant threat to many professions, including those related to the IT industry. There is a reason why the potential of machine-generated code was tested here. It's a significant threat that coding might one day escape our control, and code written in languages created in such an artificial way would be completely incomprehensible and impossible to verify by humans. It is already worth ensuring that the code remains “humanized”.

The main conclusion, however, is that GenAI-supported chatbots are already an incredible support for programmers' work today. Their language models support not only natural language transcription but can also be equally successful in transcribing code in multiple programming languages. This applies to both currently used languages and those that are becoming obsolete or have long been forgotten.

5.1 Research Contribution

The study focuses on programming languages, considering extinct languages, in the context of using GenAI tools to write code in these languages. The proposed approach is original and authored. It was decided to test 3 GenAI chatbots: ChatGPT, Google Gemini, and Microsoft Copilot. Due to the high relevance of the compared solutions and the high interest in the topic by many researchers, it may serve as inspiration and encouragement for further research in this area.

5.2 Limitations and Implications

It should be noted that the examined solutions are still being developed. The study diagnoses the current state as of the present time (spring 2024). The conducted study was based on verifying the capabilities of the

[7] Python, https://www.python.org

examined tools by formulating one-time queries. Although a comprehensive study in this area would be more reliable, it should still be expected that the results achieved would vary greatly over time.

References

Brief Programming Languages History (2024). Computer Science Degree Hub. https://www.computersciencedegreehub.com/brief-history-of-programming-languages/. Access: 2024-03-30

Brynjolfsson E., Li D., Raymond L. R. (2023). Generative AI at Work. DOI: https://doi.org/10.3386/w31161

Computer History: A Timeline of Computer Programming Languages (2018). HP Tech Takes /... Exploring today's technology for tomorrow's possibilities, October 15, 2018, https://www.hp.com/us-en/shop/tech-takes/computer-history-programming-languages#:~:text=1883%3A%20The%20first%20programming%20language,just%20numerical%20values%20of%20things. Access: 2024-03-30

Eisenbardt T. (2023). Intelligent Virtual Assistants and New Perspectives of Knowledge Acquisition, IN: Vol. 24 No. 1 (2023): Proceedings of the 24th European Conference on Knowledge Management DOI: https://doi.org/10.34190/eckm.24.1.1421

García-Peñalvo F.J., Vázquez-Ingelmo A. (2023). What Do We Mean by GenAI? A Systematic Mapping of The Evolution, Trends, and Techniques Involved in Generative AI, DOI: https://doi.org/10.9781/ijimai.2023.07.006

Haan K., Main K. (2024). 7 Best Chatbots (April 2024). Forbes Advisor, https://www.forbes.com/advisor/business/software/best-chatbots/, Access: 2024-04-02

History and Future Impact of Chat GPT (2023). Pop!Automation. https://www.popautomation.com/post/history-and-impact-of-chat-gpt. Access: 2023-04-06

History of Programming Languages (2008). O'Reilly, https://www.cs.toronto.edu/~gpenn/csc324/PLhistory.pdf. Access: 2024-03-30

History of Programming Languages (2019). Java Concept Of The Day , July 21, 2019, https://javaconceptoftheday.com/history-of-programming-languages/. Access: 2024-03-30

Huang J. (2024). https://twitter.com/Carnage4Life/status/1761483377365152234. Access: 2024-04-04

Introducing ChatGPT. OpenAI. https://openai.com/blog/chatgpt. Access: 2023-04-06

Kim E. E., Toole B. A. (1999). Ada and the First Computer, Scientific American, Inc., May 1999, https://www.cs.virginia.edu/~robins/Ada_and_the_First_Computer.pdf, access: 2024-03-30

Most Used Programming Languages (1965-2023), (2023). https://www.youtube.com/watch?v=kECnb_aKoyk. Access: 2024-03-30

Rebelo M. (2023). The best AI chatbots in 2024. Zapier. https://zapier.com/blog/best-ai-chatbot/, Access: 2024-04-02

Reynoso R. (2021). A Complete History of Artificial Intelligence. https://www.g2.com/articles/history-of-artificial-intelligence. Access: 2023-04-06

The History of Programming Languages. (2023). Saylor.org Academy. https://learn.saylor.org/mod/page/view.php?id=67404&forceview=1. Access: 2024-03-30

Toffler A. (1994). The Third Wave. Random House

Turing A. (1950) Computing Machinery and Intelligence. Mind. vol. LIX. no. 236. October 1950. pp. 433-460

Links

OpenAI ChatGPT, https://chat.openai.com
Google Gemini, https://gemini.google.com
Google Scholar, https://scholar.google.com
Microsoft Copilot, https://copilot.microsoft.com
OpenAI, https://openai.com
Python, https://www.python.org