Programming in Natural Language

Jaak Henno[1,*,†], Hannu Jaakkola[2,†] and Jukka Mäkelä[3,†]

[1] *Taltech, Ehitajate tee 5, 19086 Tallinn, Estonia*
[2] *University of Tampere/Pori campus, Pori, Finland*
[3] *Universty of Lapland, Rovaniemi, Finland*

Abstract

The whole 70-years history of electronic computers has been a fight between extremely incompatible languages: the machine code understandable by computer's processor and natural human languages understandable by computer users – humans. Main methods to overcame this incompatibility and advance software production have been introduction of high-level programming languages and reuse – libraries, but with increasing volumes and complexity of data software production is facing bigger and bigger problems. With advance of ChatGPT-like programs has appeared an insight that computers could understand natural language and the natural language could be used to write programs. Here we investigate how realistic this perspective is. In our paper we discuss about the evolution steps in software development and focus in the use of Large Language Models (LLM) based Artificial Intelligence (AI) systems in generating code based on the use of natural language as problem specification. However, development work still requires interaction between both an application area expert and a technical software developer in order to result in a reliable software solution.

Keywords

human-computer interface, programming languages, libraries, ChatGPT, truth on Internet

1. Introduction

With exhaustion of natural resources the most important resource for Humanity has become data. Data is produced in enormous amounts and data production grows exponentially. Investments in IT technology, especially in software are also growing exponentially. But the results of IT spending, Return of Investment (ROI) from IT spending is not impressive [1][2]. Growing number of IT projects do not produce expected results on time or within the budget and many are abandoned.

The main reason for IT problems is growing communication distance between the *problem area experts* who state problems (in natural language) and solution (program) implementers – *programmers*. Most of application area experts are not skilled programmers and the problems solved by ICT are becoming all the time more complex, e.g. currently real-time analysis of moving images (autonomous vehicles), translation of biologic codes and manipulating qubits. This requires also more complex programming constructs what are difficult for problem area experts to handle. New complex problems are also difficult to explain adequately to programmers, who are not specialists of the problem area.

A solution could be using far higher-level language for testing of solution ideas (compared to programming languages) – programming on natural language, what is doable also for problem area specialists and introduces new 'hot' areas also to programming old-timers. This paper examines (*problem statement*) the evolution of software development having finally

SQAMIA 2025: Workshop on Software Quality, Analysis,Monitoring, Improvement, and Applications, September 10--12, 2025, Maribor, Slovenia

* Corresponding author. † These authors contributed equally.
jaak.henno@taltech.ee; hannu.jaakkola@iki.fi; jumakela10@gmail.com
0009-0008-9886-4734 (J. Henno); 0000-0003-0188-7507 (H. Jaakkola).)

focus in the use of natural language in problem definition and the use of *artificial intelligence systems* for code generation. Recently, the practice of using systems based on *Large Language Models* (LLM) for software generation has rapidly become widespread. In this paper, the ChatGPT platform has been used as an example system. The evolutionary path under consideration also indicates a development in which the size of software increases sharply and the programmer's ability to control the produced code decreases accordingly.

In the following are described some ideas for implementing this new technology. In Chapter 2 is given an overview of earlier methods used to overcome the communication breach between problem initiators and solution implementers – using a bigger and bigger pieces of earlier code, libraries. In Chapter 3 is introduced a new approach – using some Large Language Model (LLM) based tools. To the tool (in presented examples was used ChatGPT 4.5) is given problem description of the problem in natural language and it returns a solution program (in Python 3) what is then locally executed. The conclusion from these examples is that it is possible to use LLM based tools for creating (small) programs, but these tools should be used only if the presented programs can be independently checked.

2. Evolution of software production technology

The root of computing problems lays in very different nature of languages understood by CPU (Central Processing Unit, the processor) and natural, human languages – English, French etc. We describe our problems in our natural language – English, Estonian, Finnish etc. Average human knows and uses ca 20000-35000 words [3]. For CPU these words should be translated to binary strings of machine language, where only two symbols are used. The whole history of electronic digital computers has been a fight to overcome this cognitive distance.

2.1. First revolution – high-level programming languages

For humans it is (very) difficult to work with machine language and to understand the effect, i.e. what the program does. Thus programmers started to add to machine language layers of languages which increasingly resembled their own, i.e. English language.

First appeared assemblers – encodings, were (arithmetic) operations were already denoted with meaningful symbols ("*add*"), memory locations were denoted as variables, moving values (assigning a value) become "*mov*" etc. In ca 10 years after first commercial computers appeared first 'real' programming languages – Fortran (Formula translator, aimed for engineers and scientists) and COBOL (COmmon Business-Oriented Language) for offices and banks. Currently there are over 600 programming languages. They have rather limited vocabulary, and the number of keywords in most programming languages is less than 50 [4]. Besides keywords program text contains also small number of operation symbols (+,*, ...) and other 'special characters, braces/brackets etc., also less than 20..30.

Thus, programming looks like a trivial exercise: put (very) small number of symbols into (very) small number of structures. But this new structure using less than 100 symbols should be logically equivalent with the structure described (often vaguely) using 20000...35000 words. Creating such enormous compression of information is a complex task, thus the failure rate of software projects is high.

2.2. The Second revolution - Libraries

The first programming courses of 1960s have grown tens of times to full study programs: 'Information Systems development', 'Cyber Security', 'Business Informatics', 'Bioinformatics' etc. Programming is taught in all technical and in many humanitarian and art specialties. This growth of programming has been made possible with sharing and reuse of code. Already in the first textbook on programming, The '*Preparation of Programs for an Electronic Digital Computer*' [5] the main topic was *libraries*.

All programming languages have a growing number of libraries. Nobody knows any more how many there are, e.g. it is estimated that Python has >200000 libraries [6], Javascript (node.js) - over 2.1 million packages [7] etc. Libraries call other libraries thus they add enormous number of LOC (Lines Of Code) to the program. Libraries contain many files, e.g. the popular Python library numpy (numeric Python) contains (in Python 3.11) 1426 files, the "Numpy Reference" is 2073 pages, "Numpy User Guide" – 658 pages[8]. Nobody knows exactly, what there is, why it is there, how it works or does it work at all.

3. The third revolution – Large Language Models

The main idea of the previous advances in programming technology was to use bigger and bigger pre-calculated functions – libraries. Large Language Models (LLM) allow to create much bigger functions which are described in natural language, thus could be created already by the application area specialists.

The programming process involves two stages. *First* the new program's idea, semantics is envisioned by problem area specialists in *natural language*; this idea is explained to programmers who must transform it to code in some programming language. Some problem area specialists can themselves create (simple) programs, but constantly growing complexity of both ends, problem areas and programming technology are reducing their number. Difficult (for humans) is the *second stage*, where programmer should know thousands of libraries and their functions, e.g. (in Python) where/when to use native Python language arrays, where numpy arrays, where pandas or dataframes etc. With recent proliferation of LLM based AI systems, which seem to understand natural language, this second stage could be given (at least partly) to computers.

Libraries are disburdening the human effort in code creation, transferring code creation effort to library. But the code in libraries is invisible, hidden; libraries are used like receipts 'write these imports, then write these commands'. Utterly useless are suggestions to use AI - "use convolutional neural networks; multilayer perceptrons, radial basis function network, probabilistic neural network" etc – if the application area specialist does not know 'why', does not know and understand, what actually happens, the results provided by AI are like God's voice to Moses from the burning bush. When presenting a task on natural language to LLM based AI system user must understand and describe the idea – what should be done, the semantics of the solution. The syntax - which libraries to use and how – remains for AI system; user can then investigate the presented solution in order to understand it, modify and improve. AI system will act like a (half) intelligent interface to the constantly growing mass of libraries what has become by their size directly unmanageable by humans.

This idea has been extensively discussed (e.g. [9],[10]) and the opinions vary. We wanted to test possibilities of *'programming on natural language'* using non-trivial *tasks in two areas*:

tasks based on locally available finite amount of data (classification of unknown data) and tasks based on gathering data from Internet. In all tests we used the same protocol: task text in natural-language was copy-pasted to the query window of the free version of ChatGPT-4.5 [11] (sometimes also a picture file).

3.1. Example: Classification

Information compression with classification is the basic task in all information handling. In Data Science (DS) is information compression often considered as a supervised learning – an Oracle (human expert) classifies first a large number of samples, the learning task is to create computer program which reasonably well mimicries Oracle's (humans) wisdom. To create such an algorithm a large set of already classified by some wise Oracle items is very one-sidedly divided (e.g. 80%-20%), the large part is used for teaching (computer, not human!), the smaller – for testing. For teaching are proposed some ready-made receipts/libraries (e.g. neural networks), which perform tremendous number of flops (floating-point operations). The totally non-transparent for human user process is called ML (Machine Learning), but actually this creating explanation for Oracle's classification. Without Oracle, there is nothing to learn.

New unclassified data appears all the time. Oracle can't be used for handling a set of unknown items; use of Oracle may be expensive and should be minimized. We wanted to create (basic) classification starting directly with unknown data, using some well-known ideas from DS i.e. the k-nearest neighbors algorithm [12]; in ML this is called unsupervised learning.

We tested this idea with the 'Hello World' example of DS – the *Iris dataset* [13], which presents 150 examples classified in 1936 by British statistician Ronald Fisher four attributes into three species. The classification is based on four measurements: the length and the width of the sepals and petals, thus does not provide any information about flowers 'inner' information – DNA, organelles etc. On the level of visual attributes irises are nowadays classified by very different attributes, e.g. beard, size (dwarf, tall [14]) and contrary to often repeated claim in ML papers "Iris flower has three species - setosa, versicolor, virginica..." there are actually there are 15 classes in standard classification [15], but some classifications list > 200 species of iris flowers [16]).

We wanted to investigate quick exploration - create a classification of iris data without any Oracle, using (only) ChatGPT. To compare results with the Fisher's classification Fisher's names of species were replaced with RGB color triples: (1.,0.,0) – setosa, (0.,1.,0.) – versicolor, (0.,0.,1.) – virginica. There are two possible approaches: bottom-up, considering relations between single data items and top-down – considering statistical properties of the data.

3.2. Iris – bottom-up approach

The commonly used bottom-up relation between data items is distance, used in multiple nearest neighbors methods. To ChatGPT window was copy-pasted the following text.

Task 1. "*Import list of items from the uploaded file iris_col_dat.py, calculate distances between list items using the first four elements of lists, create graph of items connecting items with minimal distance, color the graph nodes using the last element of the item as an RGB-triple (each channel encoded by real values from [0.,1.], e.g (1.,0.,0.)='red'); display the graph; find its connected components, sort them by the number of nodes in the component, print the list of components and display all components starting with largest.*"

To reduce the number of components the program created by ChatGPT was modified connecting nodes with distanece < 1.5*min_dist (Figure 1).

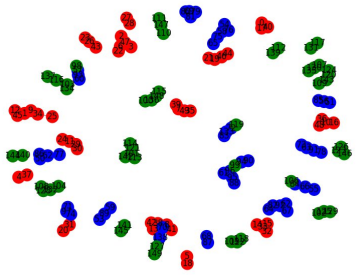

Figure 1. The graph of 42 connected components created by nearest neighbours of 150 samples of irises; alll connected components of the graph contained a single species

3.3. Iris – top-down approach

Considering distances between dimensions of sepals and petals leaves out their biological nature. The width/height of sepals/petals grow when the plant grows, but they always keep the same shape – they are similar vectors. In DS is often used reducing dimensionality [17]. As the initial statistics we considered angles of 4-dimensional vectors of data items with the 1-dimensional vector representing expectation of data items (measured by the cosine); properties of the array of angles were presented by the histogram.

Task 2. "*Create a Python 3 program, which imports from the file iris_col_dat.py the list items, finds the mean of the list and creates a histogram with 24 bins of angles of lists of first four components of lists with the mean; set color of 3 locally maximal bins of the histogram to 'aqua'.*"

The ChatGPT-s program had some small problems (it could not color histogram bars), but after *indication them* to ChatGPT it always agrees (very politely) and presents a correction, so after several interactions we got a program for histogram with colored local maximums. The histogram suggests that there are three separate groups among data – the three local maxima of the histogram. We added average values of data from these three hilltops to the list as virtual items 150, 151, 152 and used them as 'centers of attraction of angles' for grouping (Figure 2).

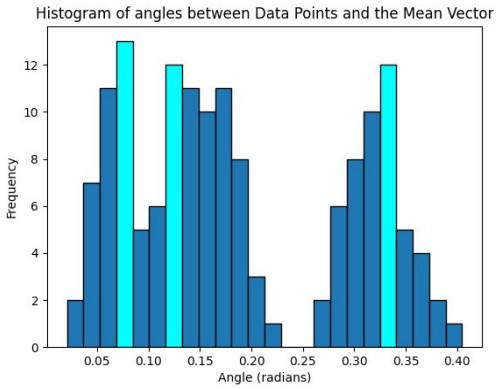

Figure2. The histogram of angles suggests that there are three separate groups among data.

Task 3. "*Create a Python 3 program, which imports from the file iris_col_dat.py the list of lists Items, using the first 4 items as coordinates finds for each list except the last three distances of the list with the last three lists, finds the minimal distance among these three distances, creates a graph whose nodes are lists from the list Items; for each node except the last thee add edge to node in last three nodes which had minimal distance with the node; color nodes using the last item in the list as the RGB triple for node color; color components are encoded as real numbers from [0.0,1.0], e.g. (1.0,0.,0.) is red.*"

After minor adjustments the result was a beautiful graph (Figure 3).

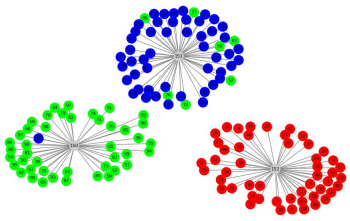

Figure 3. The graph produced by the ChatGPT program. The local maxima in histogram of angles predicted quite well specis of iris samples; experimenting with number of bins in histogram may produce even better results.

This classification agrees rather well the Fisher's one – variances $\sum_{1}^{3} E[(X_i - E(X_i))^2]$ of clusters are 0.71, 0.82, 0.49, similarity of Fisher's clustering and clustering on Figure 5 is 0.73 %.

In DS publications the Fisher's clustering is often considered as the 'final truth' - the best possible clustering of the iris data. This view is questionable, different clustering methods create different results, different methods for estimating quality of a clustering also give different estimates, it is impossible to say what is "the best", e.g. for the iris data has been proposed a clustering with four clusters [18]. Assessment of clusters produced by the well-known DS algorithm K-Means [19], the clustering produced using ChatGPT (shown in Figure 3) and Fisher's clustering using three popular methods: the Davis-Bouldin-Index (DBI, lower is better), the Silhouette score (SSC, 1 is the best, -1 is the worst), the Calinski-Harabasz Index (CHI, higher is better) gives to ChatGPT clustering the best scores (Table 1).

Table 1
Assessment of clusters

Estimator	K_Means	ChatGPT	Fisher
DBI	0.666	0.974	1.067
SSC	0.551	0.397	0.381
CHI	561.6	205.2	191.3

The classification created by ChatGPT is close to the Fisher's classification and evaluated even to be better than Fisher's.

3.4. OCR – measuring amount of obtained information

Another example tested with ChatGPT was (simplified) OCR (Optical Character Recognition), using black/white printed digits. Here attributes for classified samples are darkness values (probability p_{ij} of black pixels) in cells of uniform $n \times m$ grid drawn over the minimal bounding box of the digit. If the whole information of a digit as 1, then every cell of a $n \times m$ grid can provide $1/nm$ of the whole information. A cell does not provide any information if there is equal number of black and white pixels, i.e. $p_{ij} = 0.5$; it gives more the closer this probability is to 0 (cell is white) or to 1 (cell is black); the overall information provided by grid G_{nm} is $I(G_{nm}) = (\sum_{i=1}^{n} \sum_{j=1}^{m} (p_{ij} - 0.5)^2) / (n*m)$.

To create a grid over a single digit and illustrate effect of randomizing black pixels in grid cells to ChatGPT was uploaded image (Figure 4) and given the following parametrized task.

Task 4. "*Create a program which finds in the uploaded image minimal bounding rectangle around connected black blob, divides them with nxm grid, calculates for each grid cell the probability of black pixels in the cell, creates and prints a nxm-element vector of blackness probability values in blob's grid cells and draws next to blob (to right) similar grid where each cell is uniformly (randomly) filled with the same number of black pixels as in the corresponding cell in the grid over the black blob; show result with n=6, m=6.*"

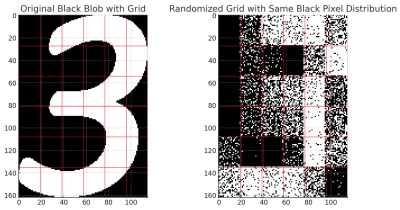

Figure 4. Uploaded image (on the left) with 6x6 grid and the grid with randomized black pixels, the first column of the 6x6 matrix of blackness values was [0.0, 0.2553606237816764, 0.7855750487329435, 0.847953216374269, 0.9376218323586745, 0.5907407407407408,...]; $I(G) = 0.309$ - i.e. ca one third of the whole information in the image

To recognize an object (a digit) in a picture, in the picture are found blobs (connected areas) of connected black pixels, their areas divided with similar grids and in every grid cell recorded the percentage of black pixels. The grids on uploaded image of digits '0','1','2','3','4' and the 5x24-matrix of darkness values in grid's cells were produced with the ChatGPT task 5 (Figure 5).

Task 5. "*Create a program which finds in the above image numbers.png minimal bounding rectangles around connected black blobs, divides them with 6x4 grid, calculates for each grid cell the percentage of black pixels in the cell and outputs 24x5 matrix where each row is a 36-element vector of blackness values in blob's grid cells; save the matrix as text file 'numbers6_6.txt' in the current directory; show on the screen the picture with applied grid.*"

Figure 5. The 6x4 grid on images of digits; grids (from left to right) contained 50.09, 47.22, 47.38, 45.54, 42.77 percent of the whole information of the corresponding digit.

The matrix was used to recognize digits on uploaded picture with task 6.

Task 6. "*Create a program which finds in the above image numbrid.png minimal bounding rectangles around connected black blobs, divides them with 6x4 grid, calculates for each grid cell the percentage of black pixels in the cell and creates for each blob a 24-element vector of blackness values in blob's grid cells, then calculates distances of this vector with three 24-element row vectors from the 5x24 matrix in the uploaded file, finds among them the minimal distance, writes to blob as a caption the index of the list giving minimal distance and shows image with captioned blobs on screen*"

The program created by ChatGPT was able to recognize digits of different size and shape (Figure 6).

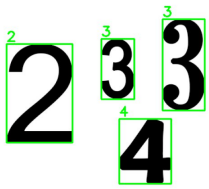

Figure 6. Program was able to recognize in an image digits of different size and shape (font).

3.5. Decoding genetic sequence

Genetic code and DNA are introduced already in primary school, thus we tested elements of these concepts also with ChatGPT. For decoding is needed information about codons (three-letter groups of nucleotides), thus to ChatGPT was uploaded the codon chart which is often used in decoding examples (Figure 7).

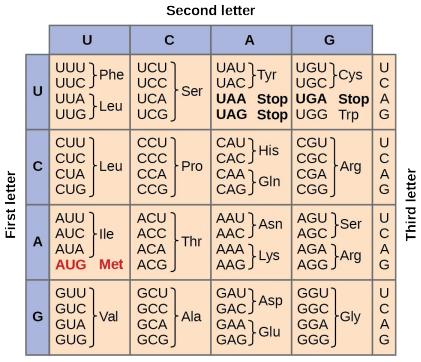

Figure 7. The codons translation chart [20]

Task: "*Using the genetic code provided, determine the amino acid sequence coded for by the DNA sequence 3' A C A T G G A A G 5"*

ChatGPT-4.5 provided solution with detailed explanations "*Reverse it... Use the codon chart you provided... Final Amino Acid Sequence: Leu - Pro – Cys*". But the process was a 'little bit too good' – the codons chart is a picture, if ChatGPT used it should have good OCR capabilities. Thus we modified the picture a bit – covered one square (Figure 8).

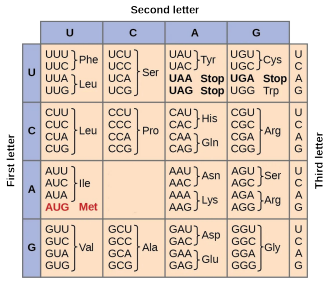

Figure 8. The modified codons translation chart

With the same task as in previous example ChatGPT presented a bit different explanations and a totally different solution.

"*Let's determine the amino acid sequence using your updated codon table:... Use the codon chart ... Final amino acid sequence: Cys – Thr – Phe*"

The program understood when all squares of the table were covered and proposed to restore the table. With partially damaged table the hidden information was obtained by some other way, but the program still claims, that it is using the provided (damaged) chart. The program could use 'outside' information (without explanation) the same way with every task – this is a very dangerous feature. Since chatGPT did not produce a working Python program for its results but used unknown and unexplained information this task was not considered in the summary table 2 (In sub-section 3.8)..

3.6. Tasks with data from Internet

We tested several LLM based AI systems with tasks involving use of data from Internet.

Task 7: "*Create a Python program which illustrates graphically change in percentage of programmers in the whole World population during the last ten years and add second-order polynomial interpolation; use real data and indicate data sources."*

The result is seen in Figure 9.

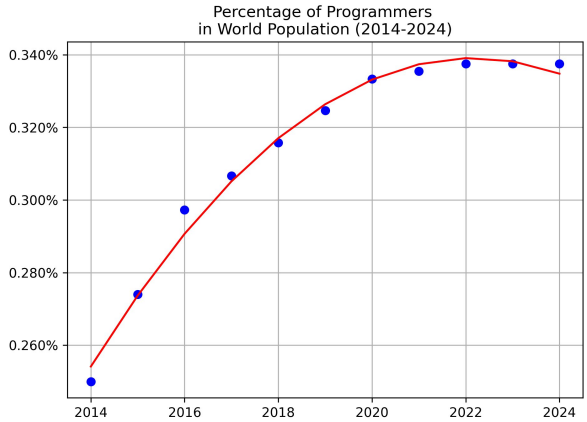

Figure 8. Graph produced by the Python program created by ChatGPT in response to task 7; ChatGPT claimed, that it used data from [21],[22],[23], but on repetitions were created different graphs.

We tested also another task where it was requested to use real data and indicate its source.

Task 8. "*For the list of countries in the uploaded file countries.txt find for each country the percentage of programmers in its population using real data from Internet, order countries by this number and create bar chart showing percentage of programmers in population in each country; calculate correlation of number of programmers with GDP per capita of countries and indicate sources of data*"

The program produced by ChatGPT created the following graphs (Figure 10 and 11) using data from [24],[25].

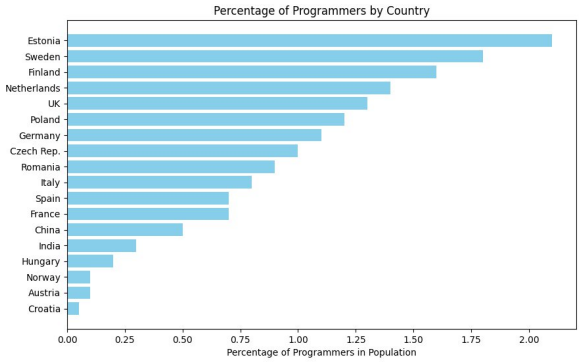

Figure 10. Graph produced by the program which was created by ChatGPT as a response to Task 8.

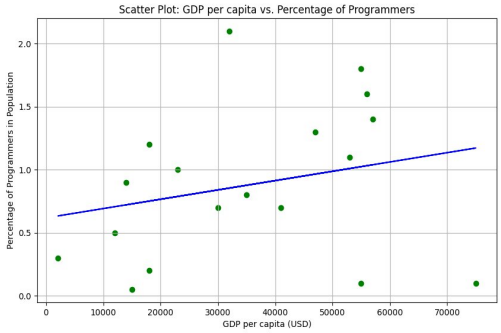

Figure 11. Another chart produced by the previous program - correlation between percentage of programmers and GDP per capita: 0.25 – doubling the percentage of programmers in population makes a country more productive.

On different queries, ChatGPT produced (slightly) different charts (although indicated the same data sources) but very different correlation coefficients, from -0.42 to 0.25.

3.7. Other LLM based AI systems

We tested besides ChatGPT also some other LLM based AI systems, e.g. the Deepseek [26], but results were worse. Common to all systems was warning (in small print) "*AI responses may be inaccurate, please verify information independently*", e.g. Claude from Anthropic repeats this warning 4 times. Nowhere are any suggestions given on how to implement 'independent verification'. When a LLM based AI system is used to generate a program, this is easy – run the program in the language's compiler/interpreter, but with information-seeking the independent verification is googling.

Most of tested LLM based AI systems did not understand what is "*real data with indicated sources*". For instance, the LLM Granite 3.2 from IBM [27], which uses 12 trillion tokens, returned a solution to the Task 4 with comments: "*For this illustrative purpose, we'll use hypothetical data, as actual comprehensive, global programmer data might not be readily available.*" The Python program produced by the Granite had also other problems: a Python dictionary was called "*data in CSV format*", unnecessary conversion of the dictionary to DataFrame for "*for easy data manipulation*" etc.

Very annoying is companies' intension to get users private information, e.g. Anthropic asks for birth date and mobile phone number – this is private info and according to EU General Data Protection Regulation (GDPR) [28] nobody is obliged to share it.

3.8. Can LLM based AI systems s improve software production?

Because of great and growing importance, programmers' productivity has been discussed often, see e.g. [29],[30],[31]. Fred Brooks stated in his book "The Mythical Man-Month: Essays on Software Engineering" [32] that a professional developer will write on average 10 lines of code (LOC) per day. Using LOC as a measure of programs size is not a very adequate metric, since most of programs code is hidden in libraries [33],[34]. A more adequate metric is total number of lines of code in all libraries imported by the program. This number characterizes the work done by the program and ratio of number of words in task description with number

of lines in all loaded libraries show the effectivity of programming is natural language. The ChatGPT reduced also our time wasted for fighting with bad modules and libraries – all Python libraries contain lot of bad modules producing errors [33].

Table 2
Number of words in task description vs number of LOC in all loaded libraries in the final program

Task	Number of words in task description	LOC in all loaded libraries in Python code	Number of good/bad modules	Ratio: LOC/number of words
1	83	1175737	2426/263	14166
2	52	1293587	2426/263	24877
3	110	1175763	2426/263	10689
4	88	1175771	2426/263	13361
5	73	1175753	2426/ 263	16106
6	97	1202764	2455/266	12400
7	34	1175736	2426/263	34580
8	61	1175708	2426/263	19274
AVG	**61**	**1193852**		**18182**

Thus a word in natural language evokes over 18 thousand LOC in Python libraries (Table 2).

4. Conclusions

The amount of data handled by Humanity is growing exponentially (Figure 12).

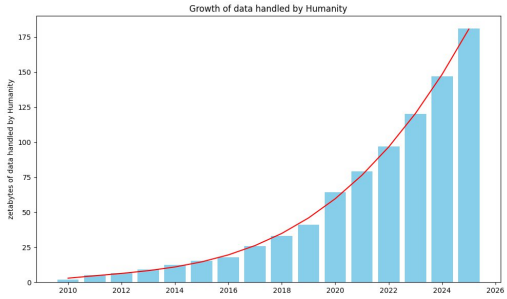

Figure 12. Growth of data (in zetabytes, 1 zB = 10^{21} bytes) handled yearly by Humanity [35]

Data is the future of economic growth [36], doubling the number of programmers in population increases country's GDP per capita by 25% (Figure 11), but 90% of data is not used [37]. The current methods of software production are not adequate, the Consortium for Information and Software Quality (CISQ) estimated that the yearly losses from poor quality software is ca 1.56 trillion USD and growing annually ca 2 % [38]. We need new low-code/no-code methods to handle our growing mountains of data [39].

Our experiments show, that it is possible to use ChatGPT (and soon maybe also some other LLM based AI systems) as a significant help in teaching and possibly also in creating new software. The use of these tools forces users to use clear, succinct language, what is not customary in our everyday practice, e.g. "clear definitions" is the first recommendation to EU Council to improve EU information systems [40]. Language improvement improves requirements, involve users, improve communication between users and developers and create better process planning and definition of final goals. Improvement in more than 16 thousand times in number of LOC in produced programs (Table 2) makes learning of new style of programming clearly a need for all application area specialists and software developers.

Our experiments show also, that LLM based AI systems would be used for solving concreate tasks with finite input – such as our tasks 1.-7., where results can be checked. Tasks, where AI systems 'wisdom from Internet' (our tasks 8.-9.) should be avoided. Quality of truth on Internet is rapidly deteriorating. More and more people publish on Internet, repeating half-earlier published half-truths or nonsense produced by AI systems and once published, it becomes a 'source', can be referred and is becoming 'truth' for many people and such papers have appeared already on Google Scholar [41]. This constantly growing flow of misinformation is a real danger for the whole humanity [42].

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT as an example system to generate code of specific problem settings in natural language. The quality of the code produced was analyzed in the paper, not published as a part of it. For comparison purposes, Deepseek, Anthropic, Granite and Gemini were tested in a light manner. In the paper text LLM based systems were not used and the authors take full responsibility for the publication's content.

References

[1] D. Courts, Software Project Failures: Why 70% Miss the Mark. https://www.callibrity.com/articles/why-software-projects-miss-the-mark.

[2] R. Williams, Banks Are Falling Short in Their ROI on Technology. https://www.crnrstone.com/gonzobanker/banks-are-falling-short-in-their-roi-on-technology.

[3] N.L. Huld, How Many Words Does the Average Person Know? https://wordcounter.io/blog/how-many-words-does-the-average-person-know.

[4] GitHub Repository: Keywords, GitHub (n.d.), https://github.com/e3b0c442/keywords.

[5] M.W.Wilkes et al., The Preparation of Programs for an Electronic Digital Computer. Addison-Wesley 1951, 167 pp.

[6] Javapoint, How many Python Packages are there? https://www.javatpoint.com/how-many-python-packages-are-there.

[7] Node.js, An introduction to the npm package manager. https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager.

[8] Numpy Documentation, https://numpy.org/doc/.

[9] J. Warden, GitHub Copilot Research Finds "Downward Pressure on Code Quality". https://dev.to/jesterxl/github-copilot-research-finds-downward-pressure-on-code-quality-4m87.

[10] Yi-Miao Yan et al., LLM-based collaborative programming: impact on students' computational thinking and self-efficacy. Nature, Feb. 7, 2025, https://www.nature.com/articles/s41599-025-04471-1

[11] ChatGPT, OpenAI, https://chatgpt.com/.

[12] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification", 1967, IEEE Transactions on Inf. Theory. 13 (1) 1967, pp 21–27, https://doi.org/10.1109/TIT.1967.1053964.

[13] UCI Machine Learning Repository, Iris. https://archive.ics.uci.edu/dataset/53/iris

[14] American Iris Society, Bearded Irises, https://www.irises.org/gardeners/care-classification/classification/.

[15] American Iris Society, Iris Encyclopedia of the American Iris Society, https://wiki.irises.org/.

[16] American Iris Society, Iris Classifications, https://www.irises.org/gardeners/care-classification/classification/.

[17] A. N. Gorban, A. Y. Zinovyev, Principal Graphs and Manifolds. arXiv:0809.0490,
[18] https://doi.org/10.48550/arXiv.0809.0490.

[19] D. Benson-Putnins et al., Spectral Clustering and Visualization: a Novel Clustering Of Fisher's Iris Data Set, https://www.siam.org/media/s12ln4i2/spectral_clustering_and_visualization.pdf.

[20] Scikit-learn Developers, Preprocessing, https://scikit-learn.org/stable/modules/preprocessing.html.

[21] OpenStax, Codon chart, https://openstax.org/apps/image-cdn/v1/f=webp/apps/archive/20230620.181811/resources/d99b2d20bb12abe0a633a49c567152957e014db7.

[22] Evans Data Corp., Worldwide Developer Population and Demographic Study 24.2., https://evansdata.com/reports/viewRelease.php?reportID=9.

[23] Stack Overflow Developer Survey 2020, Stack Overflow, https://survey.stackoverflow.co/2020.

[24] GitHub Octoverse 2021 Report, https://octoverse.github.com/2021/.

[25] Stack Overflow Developer Survey 2024, Stack Overflow, https://survey.stackoverflow.co/2024/.

[26] World Bank Group. GDP per capita, https://data.worldbank.org/indicator/NY.GDP.PCAP.CD.

[27] Deepseek, https://www.deepseek.com/.

[28] IBM, Granite Playground, https://www.ibm.com/granite/playground/app/.

[29] European Union, Data Privacy and Protection, https://www.trade.gov/european-union-data-privacy-and-protection.

[30] K. Kennedy et al. Defining and Measuring the Productivity of Programming Languages, ACM SIGPLAN Notices 18:4 (2004), pp. 441–448.

[31] J. Gmys et al., A comparative study of high-productivity high-performance programming languages for parallel metaheuristics, Swarm and Evolutionary Computation vol. 57, 2020, https://doi.org/10.1016/j.swevo.2020.100720.

[32] Y. Li et al. An empirical study to revisit productivity across different programming languages. Proc. 24th Asia-Pacific Software Engineering Conference (APSEC), pp. 526–533 (2017).

[33] F. P. Brooks, The Mythical Man-Month, Addison-Wesley (1975), ISBN 0-201-00650-2.

[34] J. Henno, H. Jaakkola, J. Mäkelä, Handling Software Icebergs, In: CEUR Workshop Proceedings, vol. 3237 (2022), https://ceur-ws.org/Vol-3237/.

[35] J. Henno, H. Jaakkola, J. Mäkelä, Non-determinism in Nowadays Computing and IT Education, In: 43rd Int. Conv. on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2020), pp. 794–801.

[36] Statista, Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028,

https://www.statista.com/statistics/871513/worldwide-data-created/ .

[37] World Economic Forum, The Future of Growth Report 2024, https://www.weforum.org/publications/the-future-of-growth-report/data-on-the-future-of-growth/.

[38] Greenenergy, 90% of data sits unused. How to get rid and avoid digital waste, https://www.greenergydatacenters.com/eng/blog/90-of-data-sits-unused-how-to-get-rid-and-avoid-digital-waste.

[39] H. Krasner, Cost of Poor Software Quality in the U.S.: A 2022 Report, https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/.

[40] T. Tran, Stay Ahead of the 2024 Latest Software Development Trends, https://www.orientsoftware.com/blog/latest-software-development-trends.

[41] F. König, What's wrong with EU information systems and how to fix it, https://www.delorscentre.eu/fileadmin/user_upload/PoPa2_Sept18_final2.pdf.

[42] J. Hader et al., GPT-fabricated scientific papers on Google Scholar, https://misinforeview.hks.harvard.edu/article/gpt-fabricated-scientific-papers-on-google-scholar-key-features-spread-and-implications-for-preempting-evidence-manipulation/.

[43] B. Mittelstadt et al., To protect science, we must use LLMs as zero-shot translators. Nature Human Behaviour volume 7, pages 1830–1832 (2023).