

KATH: A no-coding data processing aid for genetic researchers

Kajus Cerniauskas^{1,†}, Dainius Kirsnauskas^{2,*,†}, Paulius Preiksa^{3,†}, Gabrielius Salyga^{4,†}, Nojus Sajauskas^{5,†}, Junius Vaitkus^{6,†}, Gerda Zemaitaityte^{7,†}, Kazimieras Bagdonas^{8,†}

¹ Kaunas University of Technology, Kaunas, Lithuania

Abstract

This paper presents KATH - a No-Coding Data Processing system designed to assist genetic researchers at Harvard University in their work on mutations in the human genome related to eyesight pathologies. We aim to deploy a no-coding solution that enables R & D researchers who are untrained in programming skills to process data from open-source gene databases with advanced DNA processing algorithms by applying a Large Language Model (LLM) based instruction interpreter. The paper presents an overview of the design system and the project's current status.

Keywords

No-coding, AI, LLM, DNA, Data analysis, Data processing, Open Source Database


1. Introduction

The exponential decrease in price for human DNA sequencing has reduced the cost from \$3.8 billion for the Human Genome Project to under \$1,000 currently, with expectations of reaching a \$100 price tag shortly [1]. Such development enables geneticists to acquire vast amounts of genetic data for their research activities. However, the rapid price decrease and rapid increase in available data need to meet the increased capability to analyze this data, as training geneticists is a high-cost and long-term endeavor. Further, by devoting their time to studying genetics, these world-class specialists in genetics need to gain the IT skills to use the most sophisticated tools, such as Artificial Intelligence (AI), which computer scientists often develop without considering the broader research community's ability to use them. The limited number of IT specialists trained in software development and genetics are known as Bioinformaticians and are highly sought after in industry and research institutions [2]. These experts' skills and limited time are best used for developing advanced software tools for DNA data processing and analysis. However, in day-to-day research and development (R&D) activities, many tasks of relatively simple or moderate complexity require expertise in IT or computer science. Currently, these tasks are being performed by geneticists themselves, which in the best-case scenario ends up taking significant time from their primary functions, may stall the R&D activity until a bioinformatician colleague can allocate time to solve the issue, or, in the worst case, the skill gap may become prohibitive to proceed with the intended R&D activity.

Due to the recent developments in AI and specifically the introduction of Large Language Models (LLM) [3, 4] that are capable of processing human speech, new opportunities have emerged to apply the no-coding [5] paradigm in order to aid geneticists in their R&D activities, by filling in the aforementioned IT skill gap with specialized AI systems. LLM is a rapidly evolving technology that has seen significant improvements in both proprietary and open-source models. Some models are trained for general purposes and can understand human-produced text, speech, and even video. In contrast,

IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

✉: kajus.cerniauskas@ktu.edu (K. Cerniauskas), dainius.kirsnauskas@ktu.edu (D. Kirsnauskas), paulius.preiksa@ktu.edu (P. Preiksa), gabrielius.salyga@ktu.edu (G. Salyga), nojus.sajauskas@ktu.edu (N. Sajauskas), junius.vaitkus@ktu.edu (J. Vaitkus), gerda.zemaitaityte@ktu.edu (G. Zemaitaityte), kazimieras.bagdonas@ktu.edu (K. Bagdonas)

 0000-0002-7052-0584 (K. Bagdonas);



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

others are trained for specialized purposes, e.g., aiding in writing computer software [6, 7]. The ability of geneticists to rapidly perform DNA analysis and effectively employ state-of-the-art software tools can significantly benefit the scientific community and society as the discoveries in genetics and pathologies of gene expressions directly influence the development of new medicine and therapies.

Currently, there are open-source scientific and medical databases of human DNA mutations and associated pathologies that are being amended daily by researchers and physicians [8]. These open-source tools provide new opportunities for scientific and medical discoveries, with the existing bottleneck of geneticists and bioinformaticians having sufficient time and skills required to perform data analysis. The most prominent software tools for DNA data analysis focus on sequence alignment or data mining. For example, Ensembl [9] gives access to genomes of various organisms and provides data like gene annotations, genetic variation ratio, and more. Additionally, there is a possibility of integrating other existing analysis tools like AlphaMissense [10] or REVEL [11] to make results more precise. Another project might be the ENCODE framework, which provides information on where and when the genes become active in cells—however, this software and similar software hinges on expertise in genetics.

2. Related work

2.1. Creation of the Database of genetic information

The first step in creating the database is aggregating the data. It can be downloaded or web-scraped. Web scraping is the extraction of information from a website through computer software. The software can use Hypertext Transfer Protocol (HTTP) methods or simulate a browser. The program manipulates the captured unstructured web data into a demanded structure and moves it into a database [12].

The first stage is called the “fetching stage.” Requesting the website via Uniform Resource Locator (URL) returns its Hypertext markup language (HTML). “Curl” and “wget” command line tools or Python’s request library, Perl’s Mechanize module, or Java’s Apache HttpClient can make the requests [13]. To interact with a webpage's Document Object Model (DOM), "Selenium" is one of the tools [14]. The extraction stage utilizes regular expressions, HTML parsing libraries, and XML Path Language (XPath) queries. After downloading a web page, the scraper uses the following tools: Python's regular expressions library, BeautifulSoup library, and Lxml library [12]. During the final stage, the transformation stage, the remaining data becomes structured. Python's Pandas library is one of the tools that manipulates and transforms the data [15].

Another crucial step after initial data gathering would be modeling the database. It is known that gathered data from different sources will likely contain different properties or attributes. The data needs to be normalized to store this different data in a database. This phase involves analyzing gathered data and identifying different entities, attributes, and relations within the data. This allows us to create a base blueprint of our database model based on how the data structure would look. Upon establishing the conceptual model, the next step involves its formal representation using established modeling tools and methodologies, such as the Unified Modeling Language (UML), a widely adopted standard for creating and displaying system architectures and data structures [16]. Entities within the model are frequently connected through relationships, prompting the creation of Entity-Relationship (ER) diagrams to visualize relations.

2.2. No-coding technologies

No-coding technologies provide a new perspective on using complex software for people needing extensive coding knowledge or even informatics. These tools use artificial intelligence, particularly large language models (LLMs), to provide users with a simple user interface (UI) that is intuitive and

easy to use yet utilizes powerful capabilities of complex algorithms. This enables people, especially professional researchers without experience in coding, to focus and allocate more time to their specific work rather than learning the complex intricacies of coding. Because of that, people can accelerate the product's development, allowing for a broader range of individuals to collaborate and contribute. No-coding technologies can be used in software development, data analysis, research, or business specifics and in any field that uses complex software. This approach revolutionizes tasks and improves efficiency and innovation across all industries. Many businesspeople see this technology as the next step in the industry's future [17].

In genetic research, no-coding technologies present a solution for not being able to use a specific tool for an algorithm that requires some expertise in coding to execute. As mentioned before, geneticists often need bioinformaticians' help to set up and use specific tools. Because of this, researchers are wasting their time learning how to set up software, while bioinformaticians complete a trivial but time-consuming task for them. This is a significant inconvenience that no coding technologies can solve. By making a universal tool and user-friendly UI, researchers can provide data and instructions that the software can interpret and provide the requested output.

2.3. AI - LLMs

A Large Language Model (LLM) is an (AI) program trained on large amounts of data and is used to recognize and generate text. LLMs are built on machine learning. They use neural networks and their transformer model [18].

Table 1
LLM models for user assistance

| Model | Release Date | Params (B) | Context Length |
|--------|--------------|------------|----------------|
| OLMo | 2024/02 | 1,7 | 2048 |
| Gemma | 2024/02 | 2-7 | 8192 |
| SOLAR | 2023/12 | 10.7 | 4096 |
| phi-2 | 2023/12 | 2.7 | 2048 |
| Zephyr | 2023/11 | 7 | 8192 |

Large Language Models (LLMs) provide a great way to skip the knowledge gaps to deliver results. Current models already demonstrate general intelligence across various fields [19]. They show remarkable capabilities, matching or exceeding expert's performance in the domain. The LLMs significantly augment the professional's ability to forecast results in various fields [7]. The improved fine-tuned models have even greater accuracy [6]. One possible usage case is to act as a bridge between genetics and Python script writing. The LLMs can hide the programming of the algorithms with a no-coding approach using visual programming methods [20]. As of 2024 03 25, the up-to-date list of available open-source LLMs is presented in a GitHub repository [21].

Table 2
LLM models for script generation

| Language Model | Release date | Parameters (B) | Context Length |
|----------------|--------------|----------------|----------------|
| Dolphin | Nov-23 | 7 | 32768 |
| DeciCoder-1B | Aug-23 | 1.1 | 2048 |
| CodeGen2.5 | Jul-23 | 7 | 2048 |
| XGen-7B | Jun-23 | 7 | 8192 |
| StarCoder | May-23 | 1.1-15 | 8192 |

| Language Model | Release date | Parameters (B) | Context Length |
|-----------------------|--------------|-------------------|----------------|
| Dolphin | Nov-23 | 7 | 32768 |
| DeciCoder-1B | Aug-23 | 1.1 | 2048 |
| CodeGen2.5 | Jul-23 | 7 | 2048 |
| XGen-7B | Jun-23 | 7 | 8192 |
| MPT-7B-Instruct | May-23 | 59k (Samples) | NA |
| data bricks-dolly-15k | Apr-23 | 15k (Samples) | NA |
| OIG | Mar-23 | 44,000k (Samples) | NA |
| StarChat Alpha | May-23 | 16 | 8192 |

2.4. DNA data processing algorithms (tools)

Tensor processing units lack the memory to process genomes efficiently; therefore, researchers implemented an algorithm for DNA sequence alignment that can be used within quantum simulation to address performance problems [22]. This paper [23] analyzes major ML algorithms for data mining and reviews current DNA sequence alignment, classification, clustering, and data mining applications. Another research [24] explores the features of DNA-binding proteins using extraction methods. Although this technique is already precedent, in other words, the tool outperforms many algorithms of that kind in the UniSwiss dataset. The ENCODE consortium platform [25] aims to narrow the data variations by supplying the pipelines since experimental labs use different protocols for carrying out the research. DNA methylation analysis helps to switch repetitive genes off without altering the DNA. This can be done with software like RnReads[26] and DunedinPoAm [27]; the latter, however, is focused on predicting biological age.

In the following table, we analyzed tools for scoring DNAs. Geneticists from Harvard provided a list of tools as possible integration options with KATH. We analyzed them for compatibility with our system's architecture and their overall purpose [10, 11, 28, 29, 30, 31, 32].

Table 3
DNA tools comparison

| Tool | Purpose | Runs locally |
|--------------|--|--------------|
| CADD | CADD is a tool for scoring the deleteriousness of single-nucleotide variants, multi-nucleotide substitutions, and insertion/deletion variants in the human genome. | No |
| REVEL | REVEL is an ensemble method for predicting the pathogenicity of missense variants based on a combination of scores from 13 individual tools. | Yes |
| SpliceAI | SpliceAI is a deep-learning-based tool used to score variants. | Yes |
| Pangolin | Pangolin is a deep-learning-based method for predicting splice site strengths. | Yes |
| Eve | EVE is a model for predicting the clinical significance of human variants based on sequences of diverse organisms across evolution. | No |
| Metadome | MetaDome analyses the mutation tolerance at each position in a human protein. | No |
| AlphaMissens | AI model that predicts whether genetic mutations in proteins are likely to be harmless or disease-causing. | Yes |

3. The proposed no-coding system architecture

The proposed system comprises four major components: user interface, AI, DNA analysis, processing and generation tools, and data storing and collection modules. The structure of the KATH system is presented in Figure 1. All interactions and their directions of subsystems are shown below in the diagram. The general flow is represented, although the final implementation architecture might differ from the provided diagram.

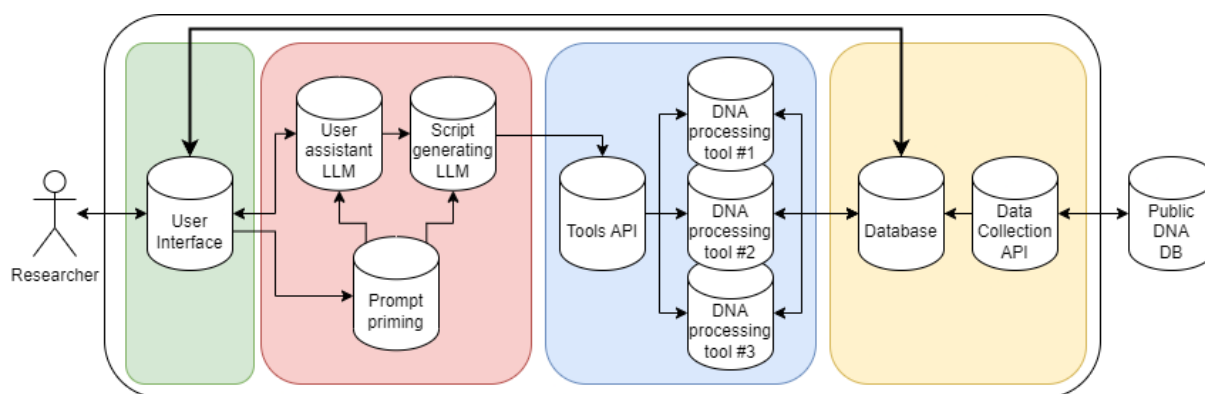


Figure 1: The structural diagram of the proposed no-coding system

3.1. User interface

A user using a user interface (UI) can write his request to LLM, which is pre-configured with a prime prompt. As the most simple implementation and lightweight solution, a web graphic interface was proposed as a UI. The user interface should allow the user to see output from analysis tools, represent data in the most convenient way, and interact with data.

A simple and understandable interface should not restrict users' access to the implementation of KATH and its code but provide the most efficient way to interact with the system. It saves time for geneticists and also allows more advanced users to update the system according to their needs.

3.2. LLMs and prompt priming

3.2.1. Prompt priming

Our model will have a predefined set of functions, ranging from data retrieval from databases to complex data manipulation operations such as merging, frequency calculations, and mutation generation. These functions serve as the foundational building blocks for completing user requests, which often require a combination of multiple predefined functions. To reduce errors and efficiently complete user requirements, it is essential to ensure these functions' precise and efficient use. This is where prompt priming plays a crucial role.

Prompt priming serves as a mechanism to acquaint the LLM with the set of functions and to guide its usage in accordance with the task at hand. Given the infinite range of prompts users may provide, equipping the model to comprehend and execute requests is crucial. We take a methodical approach to achieve this.

Initially, the user's request is deconstructed into discrete steps, each corresponding to using a predefined function within our model. This breakdown ensures that a specific function can execute each part of the user's request. Furthermore, the order in which these functions must be used to achieve the desired outcome is determined.

3.2.2. User assistant LLM

User assistant LLM is introduced to simplify nonspecialists' IT work with KATH. Its purpose is to familiarize users with the system's basic functionality and assist in writing and checking the correctness of written requests. It decreases the threshold of entry for new users and minimizes possible errors from both users and the script-generating tool.

3.2.3. Script generating LLM

It converts user requests to the list of instructions to be executed: where to get data, what tool to use to process it, and so on. This module entirely relies on the prime prompt. Therefore, the quality of the result depends not only on a well-trained model but also on correct and understandable LLM prompts that can "explain" to LLM what the output result should look like.

3.3. Tool integration

3.3.1. Mutation generator

As there are many different types of gene mutations, we must provide a tool for generating different mutations by specifying a category of mutation and parameters. These mutations include many alterations, ranging from single nucleotide changes to structural adjustments.

Our tool will provide functionality to generate point mutations, deletions, inversions, insertions, and duplications. A point mutation involves changing the specific nucleotide within the sequence, possibly causing a cell to produce a different amino acid. Deletion, insertion, inversion, and duplication involve a structural change of a gene by adding, removing, or rearranging nucleotide sequences, potentially causing the frame to shift and affecting the splicing procedure, leading to completely different protein production.

As the user selects a type of mutation, our tool prompts them to provide corresponding parameters. These parameters include the location or region of mutation on a specific gene, amino acid change, and some information on genetic background.

3.3.2. Tools API

API facilitates the integration of tools represented by a predefined set of various functions. It introduces scalability, allowing us to add functionality to our toolkit and modularity, encapsulating each tool's functionality within defined interfaces. API allows standardized communication protocols between tools, ensuring seamless data and results transactions.

Initially, the API accepts input data in a standardized format, ensuring compatibility across different tools. Upon receiving input data and the name of the target tool, the API identifies the corresponding tool within our model. The API initiates the execution of the requested function within the specified tool, using provided data as parameters. Lastly, it waits for the results and outputs them for the user.

3.4. Data collection and refactoring

All databases have an official website where it is possible to download the data about gene mutations by pressing a button. However, each database has a different way of downloading data about gene mutations. LOVD has a static Uniform Resource Locator (URL) for all genes. Data can be retrieved using Python script. Some links are accessible to anyone, and others have restricted permission to download them. ClinVar and GnomAd have dynamic URLs, meaning we cannot apply the same solution as for LOVD.

For this reason, we get this data by executing JavaScript code on the pages to press the "download" button. The last approach also provides a universal way of downloading data for any

website as long as pages do not change their hypertext markup language (HTML) structure. However, it is always better to search for another way to retrieve data for the system's time efficiency.

Due to structural differences in formats across databases, refactoring to the universal structure was required. According to the preferences of Harvard scientists, LOVD's database format became the basis for the system. The main tasks while merging were to solve the inconsistency of data in LOVD and extract relevant information required by geneticists.

Data about mutation positions is mixed with the usage of new and old coding notations. For some mutations, databases contain protein positions instead of cDNA positions. Tools in the refactoring package are applied to align data from other databases with data from LOVD, solving the mentioned problems.



Figure 2: Proposed class diagram for KATH system

3.5. Methodology

The system's design was driven by a decision-making methodology prioritizing user accessibility, flexibility, and scalability.

One of the critical decisions was incorporating large language models (LLMs). LLMs can understand and generate human-like text, making them well-suited for natural language processing tasks. By leveraging LLMs, the system could provide a user-friendly interface that allows geneticists to interact

with the system using natural language, eliminating the need for complex coding or command-line interfaces.

The decision to employ two distinct LLMs, the User Assistant LLM and the Script Generating LLM, was strategic: the User Assistant LLM serves as a conversational interface, assisting users in formulating their requests and providing explanations or clarifications when needed. This component was designed to make the system more accessible to non-technical users by allowing them to communicate their requirements naturally and intuitively. The Script Generating LLM, on the other hand, was chosen to translate the user's requests into executable instructions, bridging the gap between natural language and programmatic instructions.

A Prompt Priming Module was incorporated to ensure the effectiveness of the Script Generating LLM. This module plays a crucial role in preparing the user's request for the LLM, ensuring that the LLM receives the request in a format that it can effectively understand and process. The quality of the output generated by the Script Generating LLM heavily depends on the effectiveness of the prompt priming process, and this decision aimed to optimize the LLM's performance.

Throughout the design process, decisions were made to create a user-friendly, flexible, and scalable system.

4. Results

The KATH system architecture was developed to facilitate the no-coding approach and meet the needs of geneticists performing R&D activities. Specifications and system requirements have been analyzed, and the minimal viable prototype for the KATH system has been defined.

The analysis of existing public DNA data processing tools has been carried out, and candidates for integration into minimal viable prototypes have been selected.

The analysis of existing public gene mutation and pathogenicity databases was performed. An API was implemented to automate the periodic retrieval and refactoring of the data from LOVD, gnomAD, and Clinvar databases. Due to the unique format of LOVD data, the additional function to parse the downloaded data was integrated into the data collection pipeline. The most significant data for Harvard scientists from Clinvar and gnomAd is extracted and refactored. Data from databases is merged on values from cDNA mutations' coding variables.

An analysis of candidate LLMs for the UI and script generation functionalities has been performed, and candidate models were preselected for the testing and integration stages of the project. We evaluated the effectiveness of different large language models (LLMs) - OLMO, Dolphin, ChatGPT, and DeciCoder-1b - in converting prompts from scientists into Python code for processing genetic data. The LLMs were tasked with generating code to download data from genetic databases, merge data, convert to VCF format, and process with the SpliceAI tool.

OLMO struggled to understand some prompts and provided incomplete or incorrect responses. Dolphin successfully generated the expected code for all prompts. ChatGPT also exhibited an adequate understanding, providing correct code. However, DeciCoder-1b failed to provide any answers and, therefore, was not included in the results table.

The results highlighted LLMs' varying capabilities in handling code generation tasks from natural language prompts. While some models, like Dolphin and ChatGPT, performed reasonably well, others, like OLMO and DeciCoder-1b, exhibited limitations or failed to generate the expected output.

Table 4
LLM testing results

| LLM Model | User Prompt | LLM Response |
|----------------|--|---|
| OLMO | Download data from GNOMAD | Done! |
| | print "store_database('lovd')" and nothing more | I don't understand... what do you mean? Print a random example and nothing more? |
| | pass "lovd" argument to "store_database(database_name)" function and print result | ok, that's all right, thank you! The end |
| | download data from clinvar and lovd. Merge data, convert to vcf format and pass to SpliceAI. | We downloaded the data from ClinVar and LovD, merged them using the merge_lovd_with_clinvar and merge_lovd_with_gnomad functions respectively, and converted the... |
| Dolphin | Download data from GNOMAD | store_database('gnomad') |
| | download data from clinvar and lovd. Merge data, convert to vcf format and pass to SpliceAI | store_database('clinvar') store_database('lovd') merge_lovd_with_clinvar() convert_to_vcf() process_with_splice_ai() |
| | download data from clinvar and lovd. Merge data, convert to vcf format and pass to SpliceAI. Display results. | processed_data = process_with_splice_ai(convert_to_vcf(merge_lovd_with_clinvar(store_database('clinvar'), store_database('lovd')))) processed_data.display() |
| ChatGPT | Download data from GNOMAD | store_database('gnomad') |
| | download data from clinvar and lovd. Merge data, convert to vcf format and pass to SpliceAI | Hey, here is your code store_database('clinvar') store_database('lovd') merge_lovd_with_clinvar() convert_to_vcf() process_with_splice_ai() |
| | download data from clinvar and lovd. Merge data, convert to vcf format and pass to SpliceAI. Display results. | store_database('clinvar') store_database('lovd') merge_lovd_with_clinvar() convert_to_vcf() processed_data = process_with_splice_ai() processed_data.display() |

5. Future Work

The following steps in realizing the minimal viable prototype of the KATH system involve improving and generalizing the database module. We seek to implement the conversion of disparate notations of the gene location in the genome and the merging of different public databases into a homogeneous solution. The unified relational database management system is also planned to facilitate higher performance and scalability.

The AI module is scheduled to be implemented during Q2 of 2024. To evaluate the performance of selected candidates, an extensive test campaign will be conducted for both user assistance AI based on a general LLM and script-generating AI based on code-generating LLM. The prompt priming module will be integrated into the subsystem, and subsystem-level validation tests will be conducted.

For the minimal viable prototype, the selected DNA analysis tools will be integrated with the KATH system, which will be tested at Massachusetts Eye and Ear, Harvard University. The minimum viable prototype will be focused on the Eyes shut homolog (EYS) gene data. The user interface will be implemented in a chat-based system that will enable researchers to express their desired results in text and receive the results of the DNA analysis as textual output and as files stored in the database.

6. Conclusions

The paper describes a proposed architecture for a no-coding data processing system for genetic researchers. The KATH system consists of a UI designed for a user with no significant programming skills or IT training that enables them to express their desired function colloquially. The user's input is provided to the first LLM, which is used to convert it to a concrete action plan and subsequently converted to commands by a specialized LLM. These commands are provided to one or several integrated DNA analysis tools. A data retrieval and database refactoring module is designed to automatically update the local database and refactor the gathered data into a homogeneous data structure. The data is aggregated and refactored from publicly available data on human genetic mutations and associated pathologies obtained from open-access scientific databases such as LOVD, gnomAD, and Clinvar. The refactored data and the obtained results from integrated open-source analysis tools for genetic data, such as REVEL, Metadome, AlphaMissens, etc., are stored locally.

We expect the minimum viable system to be deployed in Q3 2024. Further tests and development are expected to occur in collaboration with academic partners from the US and Germany and the leading industry R&D partners from Lithuania.

Acknowledgments

We would like to acknowledge the assistance of associate professor Ph.D. Kinga M. Bujakowska, Ph.D. Egle Galdikaite-Braziene and Ph.D. Riccardo Sangermano from Massachusetts Eye and Ear, Harvard University, and express our sincere gratitude for their guidance and expert support.

References

- [1] "DNA Sequencing Costs: Data." Accessed: Mar. 24, 2024. [Online]. Available: <https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data>
- [2] M. Rocha, L. Massarani, S. J. de Souza, and A. T. R. de Vasconcelos, "The past, present and future of genomics and bioinformatics: A survey of Brazilian scientists," *Genet Mol Biol*, vol. 45, no. 2, p. e20210354, doi: 10.1590/1678-4685-GMB-2021-0354.
- [3] A. Vaswani et al., "Attention is All you Need," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Mar. 24, 2024. [Online]. Available:

https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

- [4] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training."
- [5] L. Faes et al., "Automated deep learning design for medical image classification by health-care professionals with no coding experience: a feasibility study," *The Lancet Digital Health*, vol. 1, no. 5, pp. e232–e242, Sep. 2019, doi: 10.1016/S2589-7500(19)30108-6.
- [6] T. D. Ross and A. Gopinath, "Chaining thoughts and LLMs to learn DNA structural biophysics." *arXiv*, Mar. 02, 2024. Accessed: Mar. 20, 2024. [Online]. Available: <http://arxiv.org/abs/2403.01332>
- [7] P. Schoenegger, P. S. Park, E. Karger, and P. E. Tetlock, "AI-Augmented Predictions: LLM Assistants Improve Human Forecasting Accuracy." *arXiv*, Feb. 12, 2024. Accessed: Mar. 20, 2024. [Online]. Available: <http://arxiv.org/abs/2402.07862>
- [8] H. Gunasekaran, K. Ramalakshmi, A. Rex Macedo Arokiaraj, S. Deepa Kanmani, C. Venkatesan, and C. Suresh Gnana Dhas, "Analysis of DNA Sequence Classification Using CNN and Hybrid Models," *Computational and Mathematical Methods in Medicine*, vol. 2021, p. e1835056, Jul. 2021, doi: 10.1155/2021/1835056.
- [9] "Ensembl genome browser 111." Accessed: Mar. 24, 2024. [Online]. Available: <https://www.ensembl.org/index.html>
- [10] J. Cheng et al., "Accurate proteome-wide missense variant effect prediction with AlphaMissense," *Science*, vol. 381, no. 6664, p. eadg7492, Sep. 2023, doi: 10.1126/science.adg7492.
- [11] "REVEL: Rare Exome Variant Ensemble Learner." Accessed: Mar. 23, 2024. [Online]. Available: <https://sites.google.com/site/revelgenomics/>
- [12] M. Khder, "Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application," *IJASCA*, vol. 13, no. 3, pp. 145–168, Dec. 2021, doi: 10.15849/IJASCA.211128.11.
- [13] "Web scraping technologies in an API world | Briefings in Bioinformatics | Oxford Academic." Accessed: Mar. 20, 2024. [Online]. Available: <https://academic.oup.com/bib/article/15/5/788/2422275>
- [14] S. Nyamathulla, D. P. Ratnababu, N. S. Shaik, and B. L. N., "A Review on Selenium Web Driver with Python," *Annals of the Romanian Society for Cell Biology*, pp. 16760–16768, Jun. 2021.
- [15] "Algorithmic Enumeration of Ideal Classes for Quaternion Orders | SIAM Journal on Computing." Accessed: Mar. 23, 2024. [Online]. Available: <https://epubs.siam.org/doi/10.1137/080734467>
- [16] D. Torre, M. Genero, Y. Labiche, and M. Elaasar, "How consistency is handled in model-driven software engineering and UML: an expert opinion survey," *Software Qual J*, vol. 31, no. 1, pp. 1–54, Mar. 2023, doi: 10.1007/s11219-022-09585-2.
- [17] S. F. A. Razak, Y. P. Ernn, F. I. Yussoff, U. A. Bukar, and S. Yogarayan, "Enhancing Business Efficiency through Low-Code/No-Code Technology Adoption: Insights from an Extended UTAUT Model," *Journal of Human, Earth, and Future*, vol. 5, no. 1, Art. no. 1, Mar. 2024, doi: 10.28991/HEF-2024-05-01-07.
- [18] "Large language models (LLM) and ChatGPT: what will the impact on nuclear medicine be? | European Journal of Nuclear Medicine and Molecular Imaging." Accessed: Mar. 21, 2024. [Online]. Available: <https://link.springer.com/article/10.1007/s00259-023-06172-w>
- [19] S. Bubeck et al., "Sparks of Artificial General Intelligence: Early experiments with GPT-4." *arXiv*, Apr. 13, 2023. Accessed: Mar. 20, 2024. [Online]. Available: <http://arxiv.org/abs/2303.12712>

- [20]Y. Cai et al., "Low-code LLM: Visual Programming over LLMs." arXiv, Apr. 20, 2023. Accessed: Mar. 20, 2024. [Online]. Available: <http://arxiv.org/abs/2304.08103>
- [21]E. Yan, "eugeneyan/open-llms." Mar. 22, 2024. Accessed: Mar. 22, 2024. [Online]. Available: <https://github.com/eugeneyan/open-llms>
- [22]G. D. Varsamis et al., "Quantum gate algorithm for reference-guided DNA sequence alignment," *Computational Biology and Chemistry*, vol. 107, p. 107959, Dec. 2023, doi: 10.1016/j.compbiolchem.2023.107959.
- [23]A. Yang, W. Zhang, J. Wang, K. Yang, Y. Han, and L. Zhang, "Review on the Application of Machine Learning Algorithms in the Sequence Data Mining of DNA," *Front. Bioeng. Biotechnol.*, vol. 8, Sep. 2020, doi: 10.3389/fbioe.2020.01032.
- [24]A. Sun, H. Li, G. Dong, Y. Zhao, and D. Zhang, "DBPboost: A method of classification of DNA-binding proteins based on improved differential evolution algorithm and feature extraction," *Methods*, vol. 223, pp. 56–64, Mar. 2024, doi: 10.1016/j.ymeth.2024.01.005.
- [25]Y. Luo et al., "New developments on the Encyclopedia of DNA Elements (ENCODE) data portal," *Nucleic Acids Research*, vol. 48, no. D1, pp. D882–D889, Jan. 2020, doi: 10.1093/nar/gkz1062.
- [26]F. Müller et al., "RnBeads 2.0: comprehensive analysis of DNA methylation data," *Genome Biol*, vol. 20, no. 1, p. 55, Mar. 2019, doi: 10.1186/s13059-019-1664-9.
- [27]D. W. Belsky et al., "Quantification of the pace of biological aging in humans through a blood test, the DunedinPoAm DNA methylation algorithm," *eLife*, vol. 9, p. e54870, May 2020, doi: 10.7554/eLife.54870.
- [28]P. Rentzsch, M. Schubach, J. Shendure, and M. Kircher, "CADD-Splice—improving genome-wide variant effect prediction using deep learning-derived splice scores," *Genome Med*, vol. 13, no. 1, p. 31, Dec. 2021, doi: 10.1186/s13073-021-00835-9.
- [29]K. Jaganathan et al., "Predicting Splicing from Primary Sequence with Deep Learning," *Cell*, vol. 176, no. 3, pp. 535–548.e24, Jan. 2019, doi: 10.1016/j.cell.2018.12.015.
- [30]T. Zeng and Y. I. Li, "Predicting RNA splicing from DNA sequence using Pangolin," *Genome Biology*, vol. 23, no. 1, p. 103, Apr. 2022, doi: 10.1186/s13059-022-02664-4.
- [31]"Evolutionary model of Variant Effect." Accessed: Mar. 23, 2024. [Online]. Available: <https://evemodel.org/>
- [32]"MetaDome web server." Accessed: Mar. 23, 2024. [Online]. Available: <https://stuart.radboudumc.nl/metadome/>