KATH: A No-Coding Data Processing Aid for Genetic Researchers

Kajus Černiauskas, Dainius Kirsnauskas, Paulius Preikša, Gabrielius Salyga, Nojus Sajauskas, Jūnius Vaitkus, Gerda Žemaitaitytė, Kazimieras Bagdonas

Kaunas University of Technology
2024

Motivation

The following issues have been identified in discussion with researchers from Harvard University:

1. Complexity of the software tools;
2. Lack of programing skills among scientist;
3. Programmers lack domain knowledge in genetics;
4. There are few bioinformaticians;
5. Even high-level specialists manually refactor and analyse data;

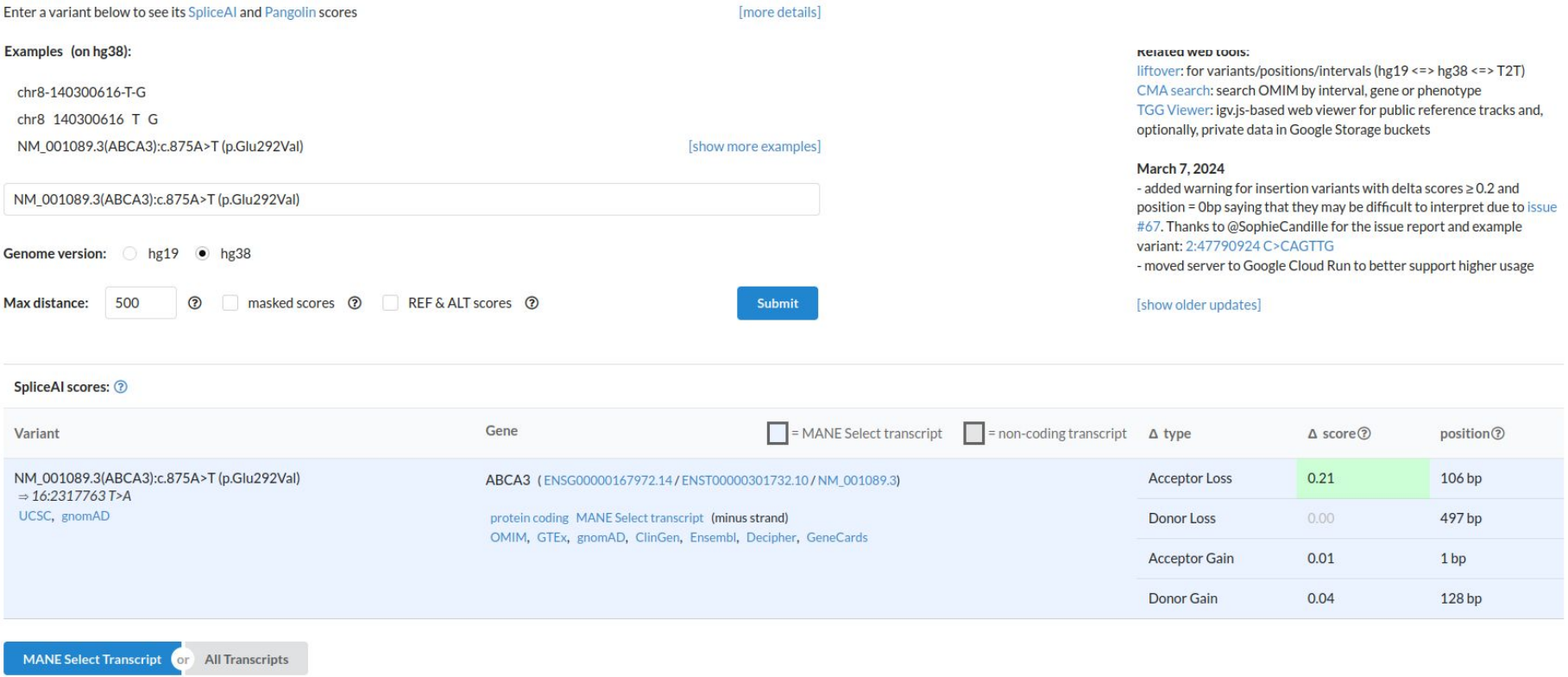
DNA analysis tools

- Existing examples of workflow automation systems:
  - NextFlow framework for creating custom workflows
  - SnakeMake for processing sequencing data with pipelines

Table 1
DNA tools comparison

Tool	Purpose	Can be run locally
SpliceAI	SpliceAI is a deep-learning-based tool used to score variants.	Yes
REVEL	REVEL is an ensemble method for predicting the pathogenicity of missense variants based on a combination of scores from 13 individual tools.	Yes
CADD	CADD is a tool for scoring the deleteriousness of single nucleotide variants, multi-nucleotide substitutions as well as insertion/deletions variants in the human genome.	No
Pangolin	Pangolin is a deep-learning based method for predicting splice site strengths.	Yes
Eve	EVE is a model for the prediction of clinical significance of human variants based on sequences of diverse organisms across evolution.	No
Metadome	MetaDome analises 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

Variation of gene analysis example



Proliferation of LLM models

Table 2
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

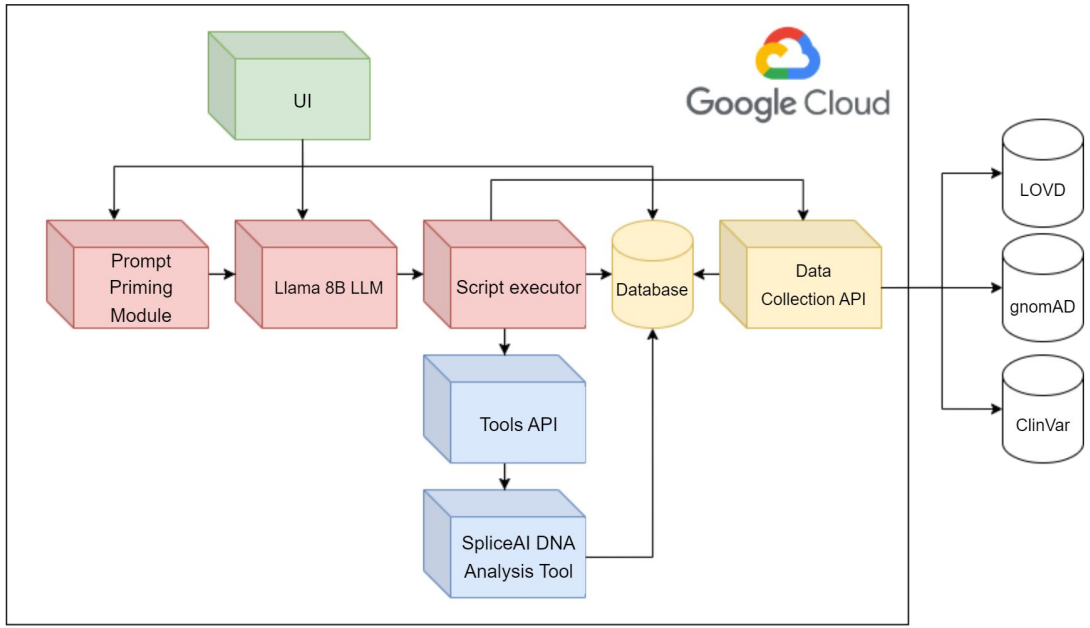
Table 3
LLM models for script generation

Table 3
LLM models for script generation

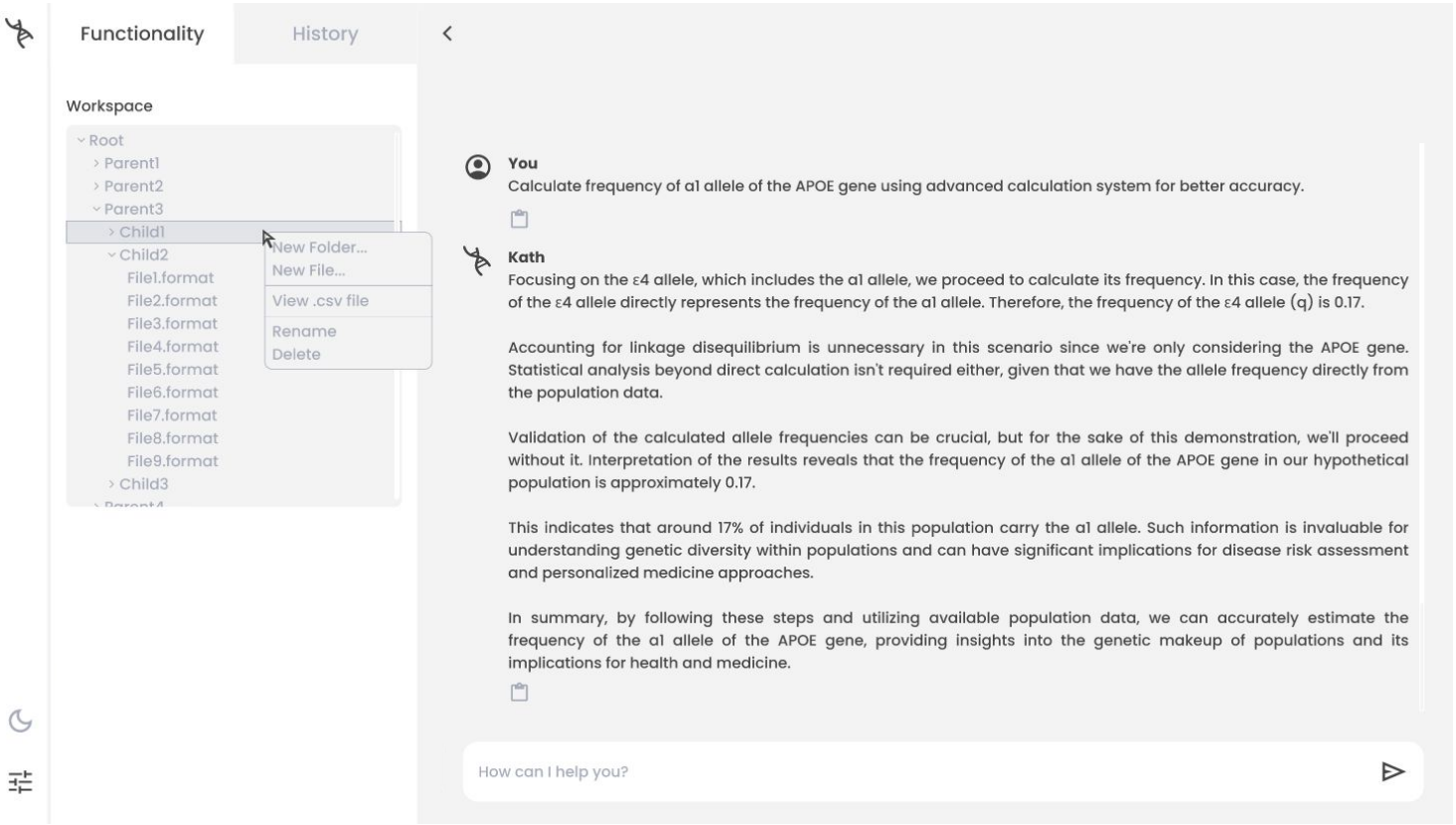
Language Model	Release date	Parameters (B)	Context Length
Dolphin	Nov-23	7	32768
DeciCoder-1B	Aug-23	1.1	2048
CodeGen2.5	Jul-23	7	2048
XGen-7B	Jun-23	7	8192
StarCoder	May-23	1.1-15	8192
MPT-7B-Instruct	May-23	59k (Samples)	NA
databricks-dolly-15k	Apr-23	15k (Samples)	NA
OIG	Mar-23	44,000k (Samples)	NA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

KATH (TRL - 4) prototype

- UI has been developed as a web service
- Llama3 8B LLM model has been employed in the KATH system
- Data collection and refactoring modules are implemented
- System is deployed on Google Cloud

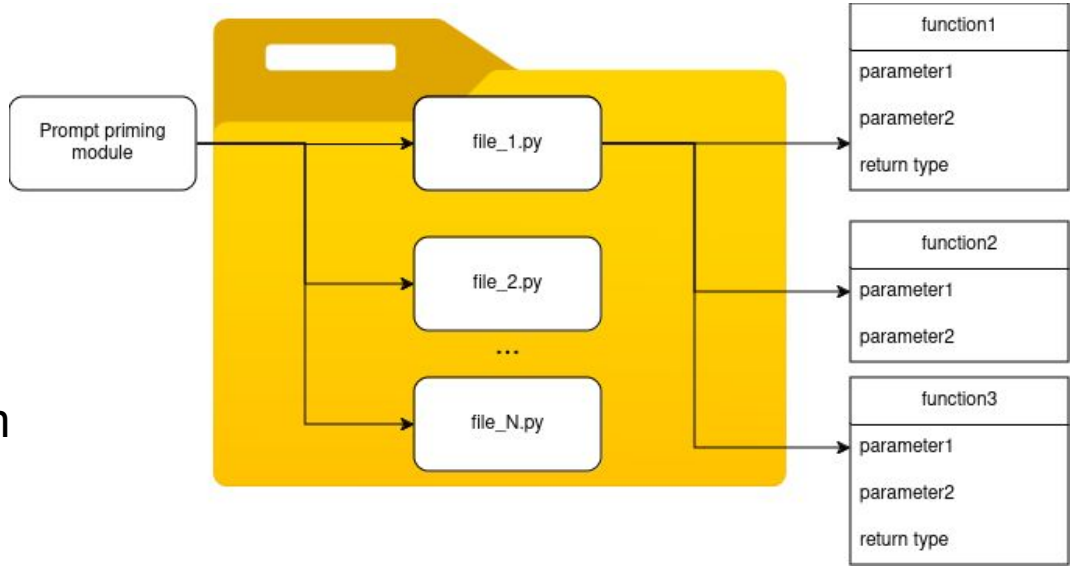


KATH system user interface



Prompt Priming Module

- Promt priming generation is automated
- System uses industry standardized documentation files from the integrated DNA analysis tools
- Custom functionality description files can be included in the system



Results 1: User query

Prime prompt:

You are Artificial Intelligence tool. Your purpose is to convert scientists prompts into Python code. You have function for downloading data from databases. When person asks you to download data use "store_database" Python function.

store_database function download data from GnomAd, Clinvar and LOVD. To download data pass name of database in lowercase. Examples: store_database('clinvar'), store_database('gnomad'), store_database('lovd'). When person asks to download data from database, print just generated function with passed arguments and nothing more.

You have "merge_lovd_with_clinvar" and "merge_lovd_with_gnomad" functions. If persons requests merging of data use this function without parameters.

You also have "convert_to_vcf" function that will convert data to vcf format. If persons requests converting to vcf use this function without parameters.

You also have "process_with_splcice_ai" function to process data with SpliceAI tool. It returns processed data. When you use this function provide variable to save data. Example, "processed_data = process_with_splice_ai". Returned object has "display" method. Use it to display data when person asks to display processed data.

Input:

Download data from clinvar and lovd. Merge data, convert to vcf format and pass to SpliceAI. Display results.

Results 2.1: Unprompted LLM output (Llama3)

```
def main():
    # Ask user for sample name, output folder, and organism
    sample_name = input("Enter sample name: ")
    output_folder = input("Enter output folder: ")
    organism = input("Enter organism (default: human): ") or "human"

    # Get Clinvar and LOVD data
    clinvar_data = get_clinvar_data()
    lovd_data = get_lovd_data()

    # Merge data
    merged_data = mergeClinvarLOVDdata(clinvar_data, lovd_data)

    # Convert to vcf format
    vcf_file = convert_to_vcf(merged_data)

    # Pass to SpliceAI
    splice_ai(vcf_file, sample_name, output_folder, organism)
```

Results 2.2: LLM output with prompt priming (Llama3)

```
def main():
    database = input("Enter database name: ")
    database = database.lower()

    if database in ['clinvar', 'lovd', 'gnomad']:
        print(store database(database))
    elif database == 'merge lovd with clinvar' :
        print(merge_lovd_with_clinvar())
    elif database == 'merge lovd with gnomad' :
        print(merge_lovd_with_gnomad())
    else:
        print("Invalid operation")
        data_from_clinvar = store_database('clinvar')
        data_from_lovd = store_database('lovd')

    merged data = merge lovd with clinvar()
    converted_data = convert_to_vcf(merged_data)
    result = process_with_splice_ai( converted_data)
    print(result)
```

Results 3.1: Unprompted LLM output (ChatGPT)

```
import pandas as pd
from pybedtools import BedTool
import subprocess

# Step 2: Merge Data
# Assuming you have merged ClinVar and LOVD data into a DataFrame called
merged data
merged_data = pd.concat([clinvar_data, lovd_data],  ignore_index=True)

# Step 3: Convert to VCF Format
# Assuming you have a function to convert DataFrame to BED format
bed_data = merged_data_to_bed(merged_data)

# Convert BED to VCF using bedtools
bedtool = BedTool.from dataframe(bed data)
vcf output = bedtool.intersect( b='/path/to/reference_genome.bed' ,
wa=True, v=True).to_dataframe()
vcf output.to_csv( 'merged_data.vcf' , sep='\t', index=False,
header=False)

# Step 4: Pass to SpliceAI
# Assuming SpliceAI is installed and available in the environment
subprocess.run([ 'spliceai', '-I', 'merged_data.vcf' , '-O',
'spliceai_predictions.vcf' ])
```

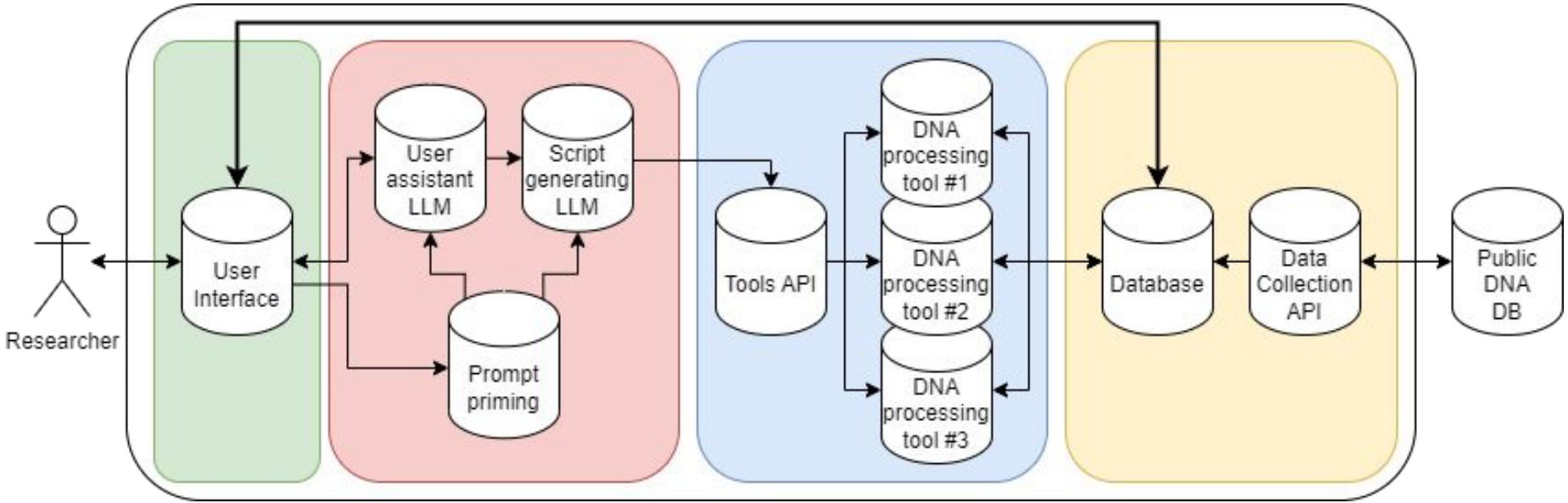
Results 3.2: LLM output with prompt priming (ChatGPT)

```
store_database('clinvar')
store_database('lovd')
merge_lovd_with_clinvar()
convert_to_vcf()
processed_data = process_with_splice_ai()
processed_data.display()
```

Results 4: SpliceAI output

```
##fileformat=VCFv4.2

##FILTER=<ID=PASS,Description="All filters passed">

##contig=<ID=6,length=63719980>

##INFO=<ID=SpliceAI,Number=.,Type=String,Description="SpliceAIv1.3.1 variant annotation. These include delta scores (DS) and delta positions (DP) for acceptor gain (AG), acceptor loss (AL), donor gain (DG), and donor loss (DL). Format: ALLELE|SYMBOL|DS_AG|DS_AL|DS_DG|DS_DL|DP_AG|DP_AL|DP_DG|DP_DL">

#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO

6   63789078   .  A  G   .   .  SpliceAI=G|EYS|0.00|0.00|0.00|0.00|3|9|-20|9
```

KATH architecture



Conclusions

1. A KATH system architecture was created for the no-coding application of DNA analysis tools and a TRL-4 prototype was developed;
2. Preliminary results prove the validity of the approach;
3. Further prompt priming research, analysis of specialized LLM models, and incorporation of additional DNA analysis tools will be performed;
4. Minimal Viable Product (TRL-7) shall be delivered to Harvard University researchers by the beginning of Q3 2024 and validation and user experience tests will be performed to fine-tune the system.

Thank you for your time!