Ayesha Binte Mostofa

- **Education:**

  B.Sc. In Computer Science And Engineering

  Bangladesh University of Engineering and Technology

  April 2019 - June 2024

  CGPA: 3.63/4.00

  Last two years CGPA [77.5 Credit] : 3.86/4.00

  Notable Courses:

  Machine Learning, Computer Security, Compiler, Computer Architecture, Microprocessors, Microcontrollers and Embedded Systems, Computer Networks, Operating Systems, Computer Graphics, Simulation and Modeling, Linear Algebra, Statistics and Probability.

- **Designation:**

  Lecturer (Full-Time)

  Department of Computer Science and Engineering,

  Canadian University of Bangladesh

  July 2024 — Present

  Past : Machine Learning Intern (Part-Time)

  Red.Digital Limited

  May 2023 — June 2023

Looking to pursue a PhD program in the field of Computer Security.

My Websites:

Ayesha Binte Mostofa (ayeshathoi.github.io)
Ayesha Binte Mostofa - Academic Coursework webpage

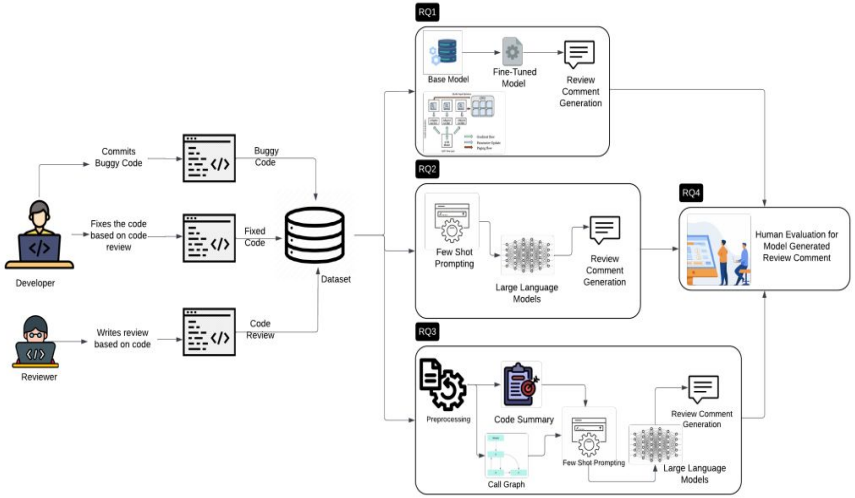
My Research

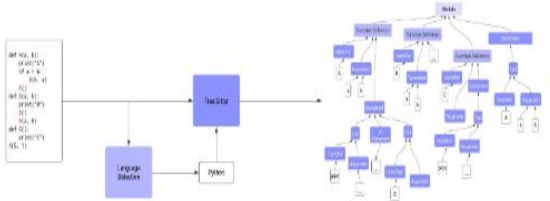
1. Advancing Code Review and Code Refinement Automation Using Large Language Models

Undergrad Thesis | Supervisors: Dr. Anindya Iqbal, Dr. Toufique Ahmed, UC Davis | Research Grant @ RISE - BUET

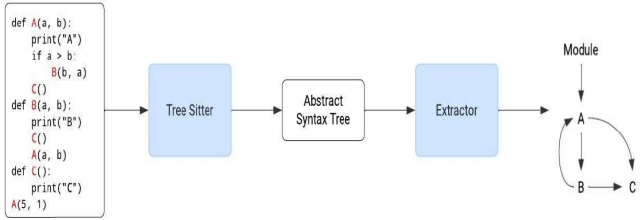
Arxiv Link : https://doi.org/10.48550/arXiv.2411.10129 | Human Evaluation Webportal

- Studied the *efficacy of* ***language models*** in **automating code review activities** by **few-shot prompting** and **parameter efficient fine-tuning** augmenting **function call graph** and **code summary**
- **Few-shot e**xamples are retrieved from training dataset using **BM25**
- QLoRA Prompt template inspired by Stanford Alpaca for fine-tuning in a supervised fashion.
- Collected the data from **CodeReviewer** on **9 languages**
- Evaluation metrics : **Bleu 4, Bertscore, Exact match (Refinement)**
- Surpassed baseline pretrained model CodeReviewer by achieving over 90% BLEU in this task.
- Paper under review for MSR 25

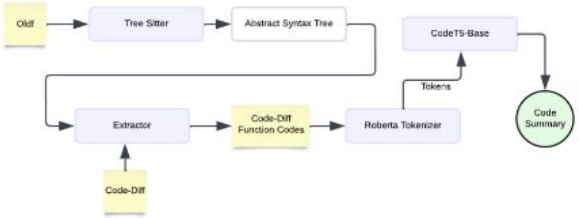




1. AST Generation



2. Call Graph Generation



3. Code Summary Generation

Dataset

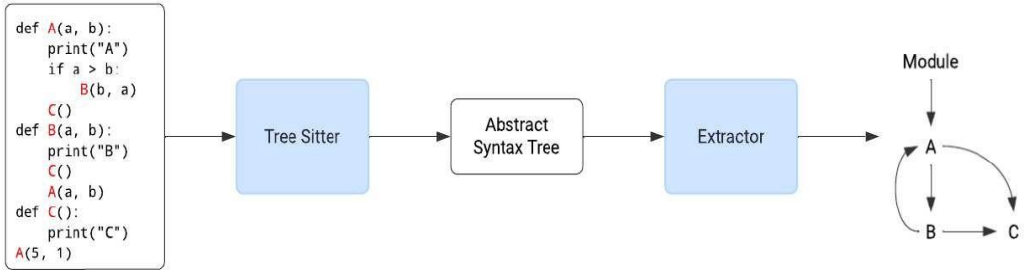
- Code Reviewer Dataset by Microsoft Research
  - Publicly available high-quality open-source repositories.
  - 9 most popular languages (C, C++, C#, Go, Java, JavaScript, PHP, Python, and Ruby)
- Augmented Code Summary & Call Graph from Data-preprocessing

TABLE I
OVERVIEW OF THE DATASET

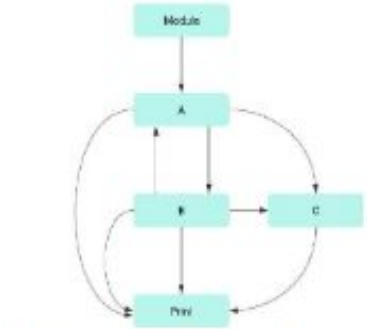
Dataset	Split Type	Count
CodeReviewer	Train Set	$\sim 118k$
	Validation Set	$\sim 10k$
	Test Set	$\sim 10k$
	Test Subset 1	5000
	Test Subset 2	500

Review Task Dataset	Patch
	msg (train)
	Call Graph (based on prompt)
	Code Summary (based on prompt)
Refinement Task Dataset	Code Before Refinement
	Code After Refinement (Train)
	comment
	Call Graph (based on prompt)
	Code Summary (based on prompt)

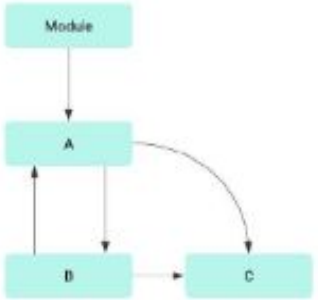
Static Call Graph Generation



Call Graph Generation Pipeline



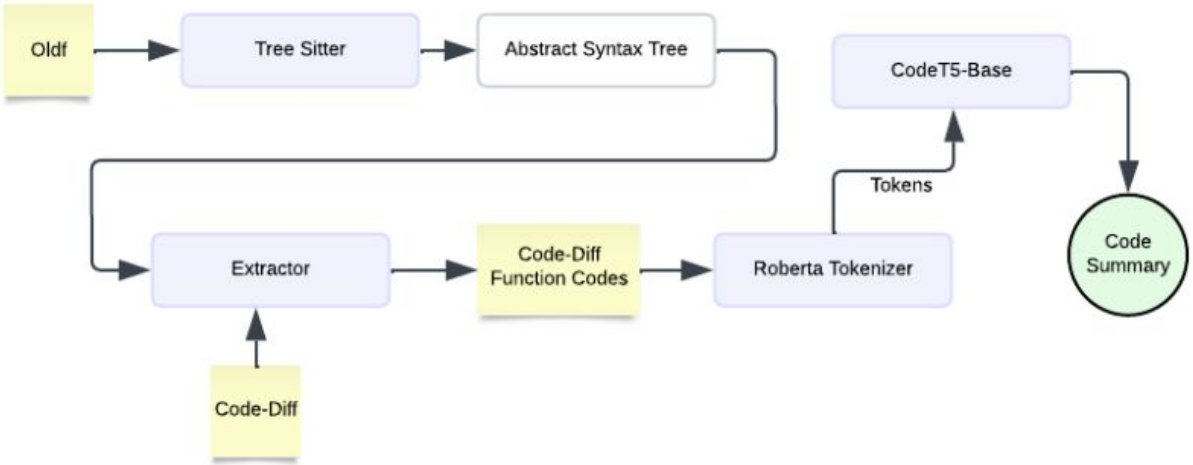
Call Graph (including built in Function)



Call Graph (in Scope)

- AST Generation Using Tree-Sitter
- Walk (DFS) on AST -> Function Calls and Class Definition Extraction
- Removal of built-in functions, scope resolution and duplication due to huge representation

Code Summary Generation



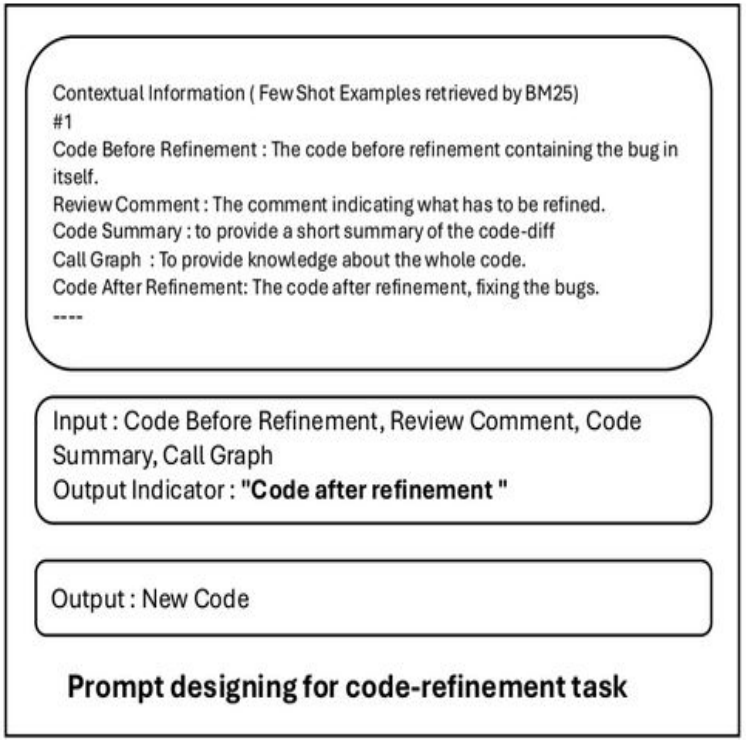
- AST Generation Using Tree-Sitter
- Walk (DFS) on the Abstract Syntax Tree -> Functions Code Extraction, mentioned in code-diff
- Summarize using codeT5-base

Few-Shot Prompting Approach Overview

Call Graph Generation by Extracting Function Call & Class Definition from AST

Creating abstract syntax tree (AST) using tree-sitter (For Structure Representation and Efficient Analysis)

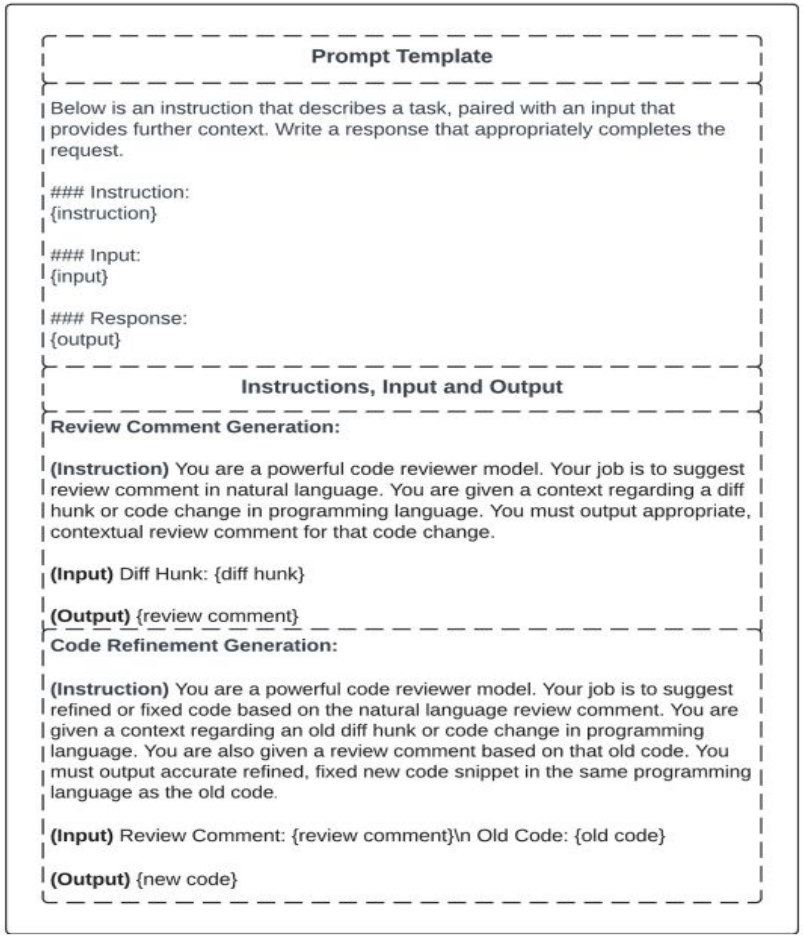
Code Summary Generation by CodeT5, with the extracted Functions Code (mentioned in code-diff)



Similarly 4 prompts for Code Refinement

QLoRA Fine Tuning Approach

- crafted our template inspired by Stanford Alpaca to perform fine-tuning in a supervised fashion by instruction-following prompt
- The modified data structure follows the {instruction, input, output} format, similar to the framework introduced in “Self-instruct: Aligning language models with self generated instructions



Results

TABLE IV
PROMPTING CLOSED-SOURCE LLMS ON TEST SUBSET 2 WITH AUGMENTED FUNCTION CALL GRAPH AND CODE SUMMARY

TABLE II
COMPARISON OF FINE-TUNED LLAMA MODELS ON TEST SUBSET 1

Model	BLEU-4	BERTScore

`@@ -185,7 +185,7 @@ Blockly.ScratchBlocks.ProcedureUtils.removeAllInputs_=function (){`
`/*`
`Blockly.ScratchBlocks.ProcedureUtils.createAllInputs_=function(connectionMap){`
`// Split the proc into components, by %n, %b, and %s (ignoring escaped).`
`-  var procComponents = this.procCode_.split(/(?=[^\\]%[nbs])/);`
`+  var procComponents = this.procCode_.split(/(?=[^\\]%[nbs])/);`
`procComponents = procComponents.map(function(c) {`
`return c.trim(); // Strip whitespace`
`});`		

Ground Truth : Is this actually a lint related change?	-	-
CodeReviewer : nit: `\s` instead of `\s`	BLEU : 0	BERTScore : 0.8094
GPT-3.5-Turbo (C) : Why do we need to strip whitespace?	BLEU : 13.89	BERTScore : 0.8758
GPT-3.5-Turbo (C+S) : Is there a reason we can't remove this line?	BLEU : 12.61	BERTScore : 0.8854
GPT-4o (C+S) : You should use a non-capturing group (?:) to improve performance, as no capturing is needed here	BLEU : 5.92	BERTScore : 0.8497
Gemini-1.0-Pro (C+S) : The code should be re-written to use a const instead of var.	BLEU : 7.81	BERTScore : 0.8507
Code Llama : `ProcedureUtils.createAllInputs_` is getting a bit hairy. Consider breaking it into several helper methods (maybe `splitIntoLines(...)` and `splitToArgParts(...)`). Alternatively, consider adding a short docstring describing it at the top, so that we can figure out what it's doing.	BLEU : 4.5	BERTScore : 0.842

Fig. 7. Model generated code review comments for a sample code diff

Validation Techniques

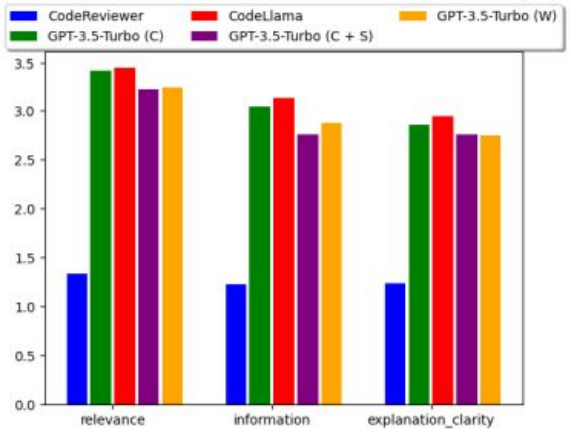
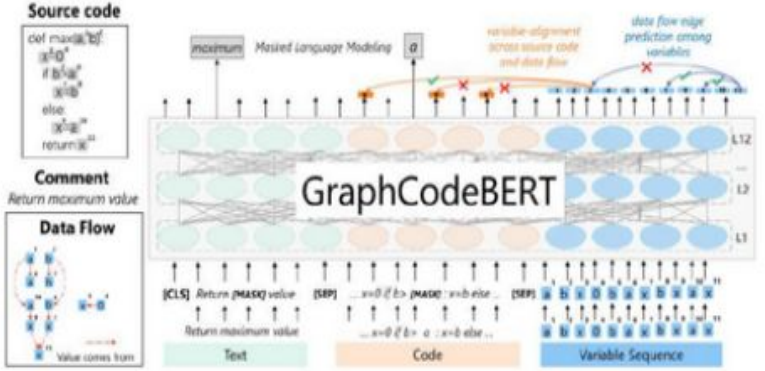


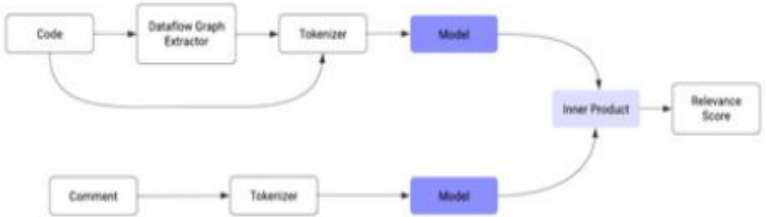
Fig. 8. Comparison of LLM-generated code reviews across their average qualitative scores on a scale of 5, as perceived by developers

TABLE VI
EVALUATION OF LLM-GENERATED CODE REVIEWS BY REAL-WORLD DEVELOPERS ACROSS THEIR AVERAGE QUALITATIVE SCORES ON A SCALE OF 5

Model	Relevance	Information	Explanation Clarity
Codereviewer	1.34	1.22	1.23
Code Llama	**3.45**	**3.13**	**2.95**
GPT-3.5-Turbo (W)	3.23	2.87	2.74
GPT-3.5-Turbo (C)	3.42	3.04	2.89
GPT-3.5-Turbo ($C + S$)	3.22	2.76	2.76



An illustration about GraphCodeBERT pre-training



Relevance Score Calculation Workflow

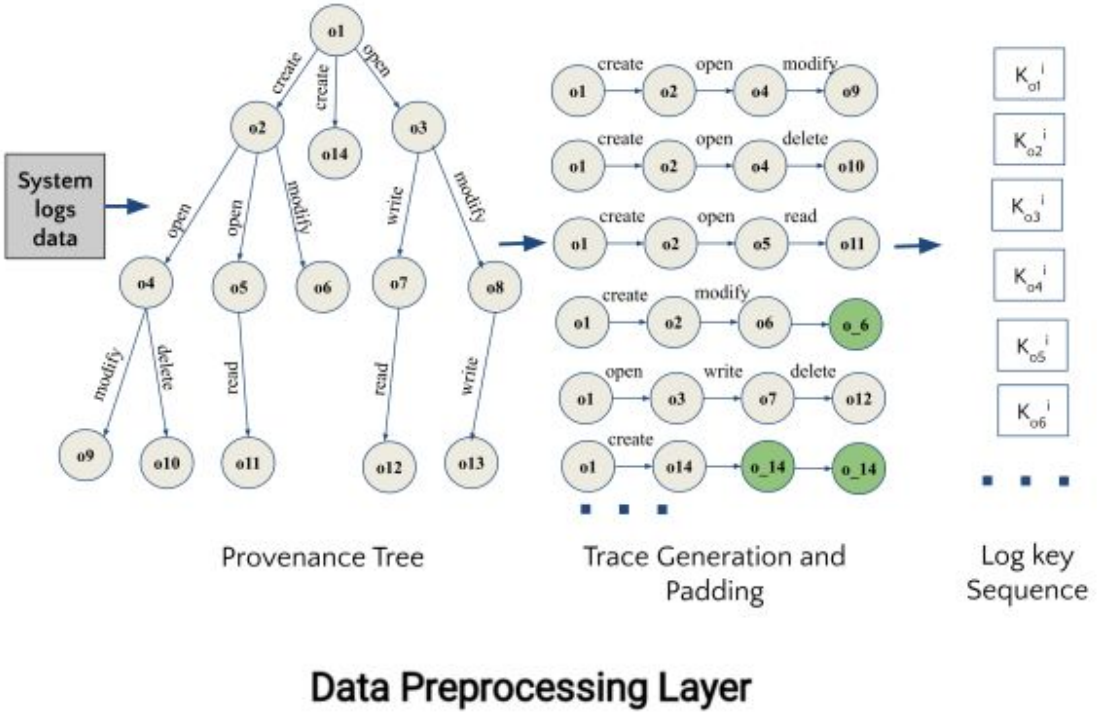
2. Effectiveness Of Language Models In Detecting Advanced Persistent Threats From System Provenance Graph

Ongoing Research Project - Supervisors: Dr. Md. Shohrab Hossain and Dr. Shahrear Iqbal, Research Officer, National Research Council (NRC) Canada

Image source: Kaspersky | APT landscape 2019

- Studied the *efficacy of* ***language models*** in detecting **advanced persistent threats** using **system provenance graphs**
- Designed a framework of creating a provenance graph from raw log data
- Collected the log data from the DARPA OPTC and DARPA TC E3 datasets
- Generated event traces from that graph
- We are planning to integrate Transformer models with RAG in our project

Data Preprocessing Layer



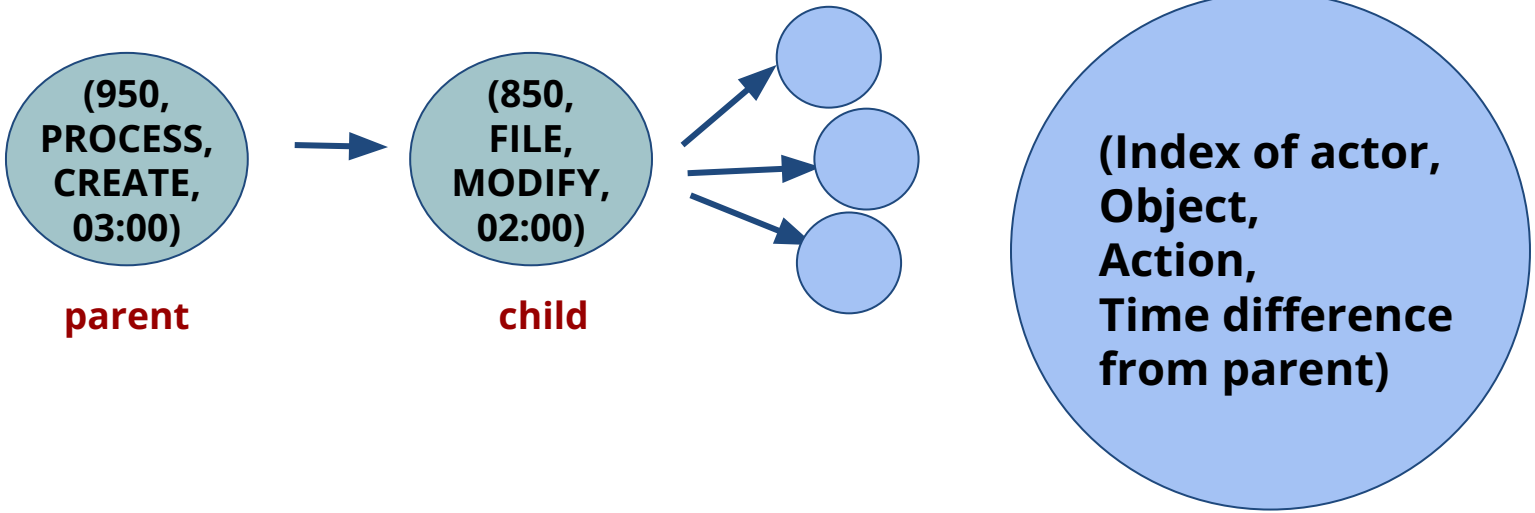
Provenance Graph Generation

parent

child

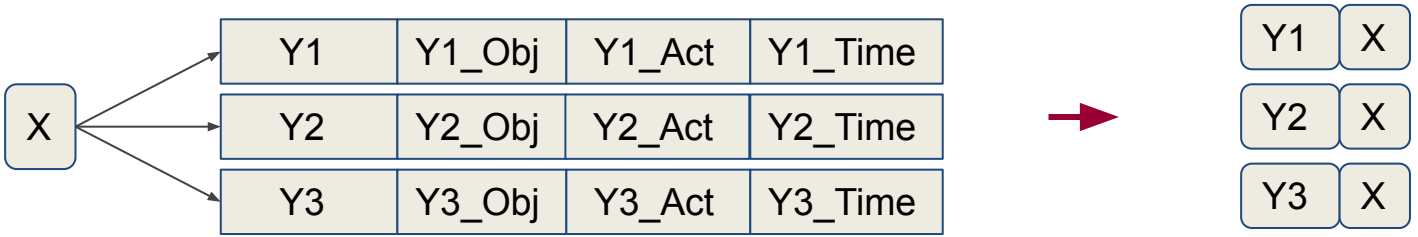
objectID	actorID	timestamp	object	action
96913629-c1c9-4503-9586-4a91de0e7311	af6b49d5-f648-41a4-946d-d92b174bae47	2019-09-23T11:23:55.857-04:00	PROCESS	CREATE
b53c1986-842c-493a-910c-78b55da2575f	96913629-c1c9-4503-9586-4a91de0e7311	2019-09-23T11:25:26.418-04:00	SHELL	COMMAND
af6b49d5-f648-41a4-946d-d92b174bae47	96913629-c1c9-4503-9586-4a91de0e7311	2019-09-23T11:25:26.416-04:00	SHELL	COMMAND

Provenance Graph Generation



Trace Generation

1. Constructing back graph



Randomly choose an event and backtrack to parent recursively, to generate a numerical trace of event ids

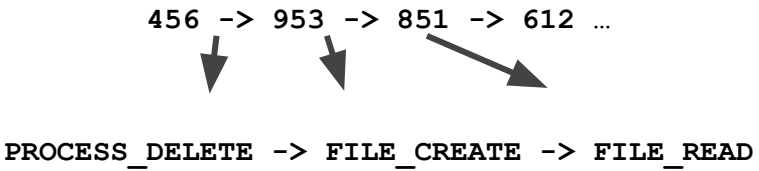
```
456 -> 953 -> 851 -> 612 ...
```

Trace Generation

2. Create a mapping of id to object action pair

id	object_action
1463	PROCESS_DELETE
1873	PROCESS_DELETE
69	PROCESS_DELETE

3. Replace event ids with object action in the traces



4. Label Malicious and Benign Data according to time stamp

Other Notable Projects

3. Paint Like your Favorite Artist : Image Style Transfer using VGG16, RESNET50 and CycleGAN [Github]

- Generated a new image by combining the content of one image with the style of another image while optimizing for minimal computational resource usage.
- Extracted features and calculate weighted style loss and content loss for perceptual loss ,
- Used FID, SSIM, and PSNR as evaluation metrics.

4. Vehicle Object Detection in context of Bangladesh Road Traffic [Github]

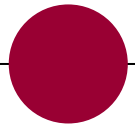
- leveraged transfer learning through advanced models such as YOLOv6, YOLOv8, rtDETR, CoDETR, Faster R-CNN.
- This project was part of the SUST Deep Learning Enigma competition in computer vision, where our team secured the 8th position among approximately 100 teams.[LeaderBoard]

5. Ground Water Level Forecasting with Machine Learning [Github]

- Groundwater levels in different regions of Bangladesh are predicted using various regression models after data preprocessing, guiding deep tube well installation.
- For seasonal predictions, ARIMA, LSTM, and GRU models are used to forecast fluctuations in groundwater levels.
- The project is funded by WARPO under the Ministry of Water Resources and supervised by Dr. A. B. M. Alim Al Islam (BUET).

My Academic Coursework

Webpage



Machine Learning Algorithms and Neural Network from Scratch

Code: https://github.com/ayeshathoi/Machine-Learning-472/tree/main

- Matrix Transformation, Eigen Decomposition & Image Reconstruction using SVD
- Logistic Regression and AdaBoost for classification
- Feed Forward Network from Scratch
- Assignment on PCA and EM Algorithm

Building a C compiler from scratch using Bison and Flex

Code: https://github.com/ayeshathoi/Compiler-310

- Symbol Table
- Lexical Analysis (FLEX)
- Syntax & Semantic Analysis (YACC/BISON)
- Intermediate Code Generation
- Intermediate Code Generation Optimization

Implementing various functionalities of the xv6 operating system

Code: https://github.com/ayeshathoi/OS-314/tree/main

- Shell Script
- Inter Process Communication
  - pthread
- xv6 operating system
  - System Call Implementation : Trace & SYSINFO
  - Process Scheduling using Lottery Scheduling Algorithm
  - Memory Management using Paging Framework

Cryptography Algorithms, Security Vulnerabilities, Attacks and Protection

Code: https://github.com/ayeshathoi/Security-406/tree/main

- **Cryptosystem (AES, RSA, Diffie-hellman & TCP-Socket implementation)**
- **Buffer Overflow**
- **Malware Assigment (FooWorm and Modification in AbraWorm)**
- **Firewall Exploration & Evasion**

VLAN Configuration and Wireless Network Simulation Experiments

Code: https://github.com/ayeshathoi/Computer-Networks-322/tree/main

- **Socket Programming**
- **Packet Tracer**
- **Simulation in NS2 (configuring mac,routing protocl etc.)**
- **Wireshark**
- **Term Project**
    - CUBIC-FIT: A High Performance and TCP CUBIC Friendly Congestion Control Algorithm
    - TCP Cubic-Fit Modification in ns2 | Report

Achievements

Honors and Awards 🏆

1. **RISE-BUET Internal Student Research Grant, November 2023** [Details] : Received 81518 BDT as grant for our undergrad thesis research from RISE-BUET.

2. **NSysS Research Poster Presentation, December 2023 [Details] :** Presented our poster with research findings at the NSysS in December 2023.

3. **SUST DEEP LEARNING ENIGMA 1.0 Finalists, 2024 (Among 100 teams around Bangladesh):** We address complex task of object detection in driving environments across 9 districts in Bangladesh using BadODD dataset. We employed transfer learning using three different approaches, such as : YOLOv6 and YOLOv8, rtDETR and CoDETR and Faster R-CNN. [Github] [LeaderBoard]

4. **Dean's List Scholarship, 2023 - 2024** : This scholarship is granted to undergraduate students for their academic excellence.

5. **Comilla Board Talentpool Scholarship**

    - **HSC, 2018 : 5th position Holder (1st Among Girls)** [Details]
    - **SSC, 2016 : 8th position Holder** [Details]

Leadership 

1. **IEEE Computer Society BUET Student Branch Chapter (2021-2023)** [Check out all of our events]
   - Publicity Committee Coordinator : 2022-2023 ; Media Committee Executive : 2021-2022
   - Helped organizing competitions, career talks, research seminars and software development workshops

2. **Online Class Recording Management (2020-2024)**

   maintained all recordings for my classmates in my youtube channel as unlisted playlists from the covid.

3. **Blogs Writing (2024)**

   Recently, I have started writing blogs sharing my experience with a view to helping others. Blog Link

4. **Bangladesh Army Commissioned Officer Selection, Long Course - 80 (2018) [Details]**
   - Day 1 : Intelligence Test, Picture Perception and Description Test, Personality Test, Essay Writing (Bangla, English), Medical Test
   - Day 2 : Group Discussion 1, Progressive Group Task, Half Group Task, Extempore Speech, Physical Ability Test, Interview
   - Day 3 : Planning Exercise (Written & Discussion), Command Tasks, Mutual Assessment, Interview
   - Day 4 : Assessment conference, Announcement of final results and Final Briefing