Projects for CS 4366

1. Chat with Github: Enhanced GitHub Repository Analysis with Large Language Models

Navigating through a GitHub repository to comprehend or modify its functions can be daunting, especially for programming beginners. Typically, one might need to delve into every function and perhaps even run it locally, dealing with environment setups, which can be quite time-consuming.

Although one can paste code snippets from GitHub to ChatGPT for explanations, several challenges arise:

Code Distribution: A GitHub repository often spreads code over multiple files and directories, making it cumbersome to ask questions about the entire codebase.

Model Memory Limit: If a code snippet is extensive, models like ChatGPT may forget earlier parts of the code, leading to incomplete or inaccurate responses.

Overlooking Issues/Discussion Content: Developers often provide valuable insights in the Issues or Discussion section, which can easily be missed.

To address this, we propose leveraging large language models to dissect GitHub repositories into manageable chunks, converting them into a vector database. When a query is posed, the model can fetch relevant code snippets and provide explanations. Additionally, it can scan the Issues/Discussions sections to provide insights based on prior discussions. Projects like chat-pdf on GitHub, which embed and match vectors to explain PDF content, can serve as a blueprint. Explore similar work here: https://www.chatpdf.com/.

2. Generating Human-Like Programming Errors Using Large Language Models

In the realm of program repair, augmenting datasets often involves introducing syntax or logical errors into correct code. Conventionally, this is done by randomly replacing, deleting, or inserting code snippets. However, these artificial errors might not closely resemble errors made by real-world programmers.

The question arises: How do researchers typically corrupt a program to expand their datasets? More information can be found at Program Repair(https://program-repair.org/).

Can large language models be leveraged to create errors that mimic those made by beginner programmers, rather than just random token manipulations? Generating "realistic" errors would greatly benefit research in this domain.

3. Smart Contract vulnerability detection with Large Language Models

Smart contracts, developed in languages like Solidity or Vyper, form a cornerstone of blockchain technology. In the last three years, there has been a surge in research, particularly in the software engineering domain, investigating the application of machine learning and large language models to smart contracts and blockchain, for example, smart contract vulnerability detection.

4. Leverage Stackoverflow and Large Language Models to Solve Latest Program Error

ChatGPT's training data is limited to 2021, and while there was once an integrated Bing search plugin, it has been temporarily suspended due to load concerns. For programming errors, many of the latest issues are discussed and resolved on platforms like GitHub Discussions and Stack Overflow. However, these solutions might not be present in ChatGPT or other LLMs (Large Language Models) due to the constraints of their training datasets. How can we leverage the latest corpus resources to detect and correct program errors or smart contract vulnerabilities?

5. Code generation from requirement

Leveraging the power of Code Llama, we will enhance its capability to seamlessly convert natural language project descriptions and/or project specifications into functional code snippets, by fine-tuning Code Llama. During its fine-tuning process, we will train Code Llama on a vast array of natural language-to-code pairs, ensuring it captures the nuances of different programming paradigms and idioms. This approach allows developers to ideate and prototype rapidly without the initial hurdle of code syntax.

6. Reengineering the project specification from its source code

Reengineering a project specification from its source code involves reverse engineering, which is the process of analyzing and understanding the existing codebase to extract a high-level specification or design documentation. This reengineering task first needs to conduct a thorough review of the source code to understand its structure, logic, and components, and identify and document the various modules, functions, and classes present in the code. Then, it will further extract information from code comments that provide insights into the purpose and functionality of different code segments. With all information to generate the project specification.

7. Code generation from flowcharts

A flowchart is a type of diagram that represents an algorithm, workflow or process. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. Code generation from flowcharts involves translating the diagrammatic representation of a program (flowchart) into executable code. It is essential to note that fully automated code generation from flowcharts is a complex task. It needs to map flowchart elements (symbols) to corresponding code constructs. For example, rectangles might represent procedural code blocks, diamonds might represent conditional statements (if-else), and arrows might represent the flow of control between code blocks. Besides, it also needs to handle the loop structures and error handlings in the flowchart.

8. Reengineering the flowchart from a source code

Reengineering a flowchart from existing source code involves analyzing the code to extract the logical structure and control flow, then representing that information visually in a flowchart. It needs to understand the source code structure, including functions, loops, conditionals, and other control flow structures, and identify the main components, such as functions or methods, and their relationships. Each function may correspond to a distinct block in the flowchart. It also needs to trace the control flow through the code, noting how decisions (if statements) and loops are structured, to identify branching points and how the program responds to different conditions, and to identify sections of code dedicated to error handling or exception handling and loop structures. These may be represented as decision points in the flowchart.

And finally, it will create a visual representation of the code by drawing a flowchart using standard symbols for processes, decisions, loops, and connectors and arrange the flowchart in a way that reflects the organization and sequence of the code.

9. Code Summarization

To make students' code submission more comprehensible and navigable, especially for newcomers or for seasoned developer performing code reviews, Code Llama will be utilized to construct a code-summarization tool to summarize intricate code blocks into clear and concise natural language descriptions. By training Code Llama on diverse code structures and their corresponding summaries, it can distill lengthy or complex code segments into easily digestible explanations. This function is particularly beneficial for both seasoned developers wanting a quick overview and programming beginners aiming to understand advanced code logic. This project needs us to explore how the intricate capabilities of the GPT model can be harnessed to generate concise and coherent summaries of extensive code blocks. By emphasizing the salient features of code structures and logic, the code summarization tool can provide developers and researchers with succinct, yet informative overviews of their codebase.

10. Utilizing LLMs to convert a program from one language to another (e.g., Jave to Python)

Converting a program from one programming language to another is a complex task that involves understanding the logic and structure of the original code and then translating it into the syntax and conventions of the target language. However, it is feasible and much easier to use LLMs to develop automatic tools for such tasks.

11. Deepfake detection

Deepfakes involve the manipulation of audio, video, or images to depict content that is not based on real, authentic data. This technology enables the creation of highly realistic and often convincing content, such as altered facial expressions, lip-syncing, or even entirely fabricated scenes featuring individuals. Deepfakes can be created for various purposes, including entertainment, political satire, or, unfortunately, malicious activities. Therefore, it is important to detect deepfakes. Detecting deepfakes is challenging. We can focus on detecting a specific type of deepfakes, such as detecting deepfake images, detecting deepfake audios, or detect deepfake videos.

12. An online animals and plants distribution system

Development of an online system to trace the distribution of animals and plants. It encourages users to report animals and plants around the place they live. Each report includes the location, the time, the name, and the images of different animals and plants, and a simple description of these animals and plants. In addition, the system can automatically correct/remind users if users provide an incorrect name for an animal or a plant. One important function of this online system is it can dynamically display the migration patterns of different types of birds.

13. A Covid sequelae system

Development of a COVID sequelae system, which monitors and manages the long-term health consequences, known as sequelae, experienced by individuals who have recovered from COVID-19.

14. Summarization of a stack of papers using LLMs

Summarizing a stack of papers involves systematically reviewing and condensing the key information from each paper to provide a concise overview. It is time consuming. To save readers' time, it might be helpful to utilizing LLMs. Leveraging LLMs like GPT-3 for summarizing a stack of papers involves using natural language processing capabilities to generate concise and coherent summaries. It might be feasible to have two stages to come out the summarization of a stack of papers. The first step is to generate a summary for each paper using LLMs, and the second step is to aggregate all summaries of the stack of papers into a cohesive document.

15. A paper submission and reviewing system

A paper submission and reviewing system, like https://cmt3.research.microsoft.com/ or https://easychair.org/

16. Development of a website for my machine learning and data science lab

The website can contain several pages, such as a home page, a people page, a research page, and a publication page, and a news page.

17. Chatbot for a computer language learning

Developing a chatbot for computer language learning involves combining natural language processing (NLP) capabilities with educational content delivery.

18. Chatbot for a foreign language learning

Develop a chatbot for helping users learn a new language with interactive lessons, quizzes, and pronunciation exercises.

19. Chatbot for learning algorithms

Developing a chatbot for learning algorithms involves integrating educational content, natural language processing (NLP), and interactive features to facilitate a user-friendly learning experience based on algorithms and code snippets in the textbooks, lecture slides, assignments, projects, and exams. It must ensure that the chatbot can help beginners to learn well a range of algorithms (sorting, searching, graph algorithms, etc.).

20. Chatbot for personal health

Developing a chatbot for personal health involves integrating health-related information, providing guidance, and potentially offering personalized assistance. The chatbot has the conversational flow to cover health-related topics such as symptoms, wellness tips, fitness routines, nutrition advice, or mental health support. Ensure that the chatbot engages users in a friendly and supportive manner. For example, it contains a symptom checker feature that allows users to describe symptoms, and the chatbot provides general information without giving medical advice and encourages users to consult healthcare professionals for specific concerns. It also contains wellness tips and recommendations, nutrition guidance, fitness and exercise plans, mental health support, and medication reminders.