Solus

Automating the development process.

CSX Labs

Table of Contents

CSX Labs

CSX Labs Overview

CSX Labs (Computer Science Exploration Laboratories) is an exploratory research organization committed to exploring, developing, and promoting cutting-edge technologies that advance humanity. More information about our organization can be found on our website and Business Plan.

Solus Project Overview

Solus aims to bridge gaps between AI and humans in software development by allowing language models to interface with the internet and filesystem, opening their contributions to developers on a wider scale. While Solus codes and debugs, developers can focus on higher-level tasks like architecture, design, and requirements. Solus will be able to generate an entire *project* from a conversation with the developer.

Research Problem

There aren't any *complete* AI project generators. Current AI-assisted software development services, such as GitHub Copilot and Replit's Ghostwriter, are limited by their lack of access to external contexts like online documentation, a debugger, or library updates outside their training data. This lack of sufficient context limits these services to assistants rather than complete development solutions.

Proposed Solution: Solus

Solus is an AI-assisted project generator designed to automate project creation from well-defined requirements for individual developers and development teams. Solus will access the same context as developers, bridging the gap between code AI assistants and complete AI project generators. Solus's key components include:

1. Chat
    a. Solus will chat with the developer, gathering requirements for the project and using its query functionality to get outside information. There are already readily-available chat-based language models, like OpenAI's gpt-3.5-turbo and gpt-4, that we can use for the developer to converse with the user.
2. Querying (Query API)
    a. A language model generates a JSON query and sends it to the Query API.

   b. Solus will integrate with search APIs, such as Google Custom Search, and other essential websites, such as StackOverflow, to gather a corpus of information about a query. We'll integrate with some common websites and databases to remove overhead from scraping. However, we will provide scraping functionality (of returned search results), so Solus can access external documentation, API specifications, and other requisite information.
   c. Solus will then use a language model to scan over the corpus and distill information relevant to the query. In the future, we plan to implement search methods to pre-align the language model with the important parts of the pages, as large language models are latent and compute-expensive.
   d. Solus will serialize the distilled information into a JSON response and send it to the requestee.
3. Project Planning
   a. Solus will generate and revise project outlines and requirements in YAML from the chat conversations and queried contexts.
   b. Solus will use this outline to generate and revise the project's folder and file structure. Solus will apply the project structure to the file system.
   c. Solus will also take the project requirements and use the Query API to get dependency information.
4. Coding & Debugging
   a. Solus will utilize language models to generate the code, requesting context from a context engine and outside information from the Query API.
   b. There will be continuous refactoring throughout the development process. We will start with regenerating the entire file to refactor, though we plan to integrate in-place editing functionality in the future.
   c. Solus' language models will have access to a Debugging API, allowing them to interface with a debugger and fix errors.
5. Context Engine
   a. Solus' context engine will enable a language model to search the entire codebase and add items to its context. Solus passes in context at the beginning of any prompt given to the model.
   b. We will start rudimentary with simple text concatenation, though we plan to include search functionality and advanced context management features in the future.

GCP Tools to be Used

Cloud Pricing Estimate - $11.69/mo - $140.28/yr
The GCP Tools here are auxiliary to our project management. Though they aren't available for complete public consumption (we haven't gotten past the waitlist yet), we would love to test-drive Google AI's language models, such as LaMDA and PaLM, through Cloud Vertex's Generative AI.

- Cloud Storage: For storing example contexts and caching scraped files (for testing).
- Cloud Build: For building binaries and artifacts of Solus.
- Cloud Source Repositories: For mirroring our GitHub repository in our build pipeline.
- Artifact Registry: For storing project artifacts.

Project Timeline

We are trying to overestimate the time needed for this project to account for possible intermittent problems. We are following the Tracer Bullets philosophy, as in we are iterating on ALL the components to complete general requirements while collecting continuous feedback.

- Month 1-4: Implement elementary requirements collection, querying, code generation, and rudimentary refactoring. (MVP)
- Month 5-9: Prepare the project for mass public consumption and user-driven development. Add dependency collection, and improve response serialization and context. This project is open-source, so the preparation process will involve setting up repository rules and reviewing policies, preparing our project for mass contribution.
- Month 10-12: Iterate on user feedback and iterate on features. Get funding.

Key Milestones and Outcomes

1. An end-to-end project generation.
    a. Outcome: A CLI application that can generate a rudimentary project from a conversation with the developer. The first projects may not be functional, though building end-to-end generation will fully actuate Solus in communicating with the filesystem and the internet.
2. Project refactoring and debugging.

   a. Outcome: Solus should be able to refactor code and interface with a debugger. We may be able to use models like OpenAI's GPT-3 Edit to achieve this.
   b. Debugging and refactoring are functionality that depends on the quality of our Context API. Therefore, our ability to complete the refactoring and debugging API will indicate the quality of our Context API.
   c. Debugging and refactoring also depend on our AI-to-machine communication, or how we communicate and interpret messages generated by language models across components.
3. Mass Adoption by Software Engineers
   a. Outcome: Solus should be affordable and accessible enough to be mass adopted by individual software engineers.
   b. We want our tool to be accessible to as many developers as possible. Improving accessibility includes improving our Terminal User Interface (TUI), iterating on the cost of project generations, and shortening the sign-up (and API access) process. We believe the best way to improve a tool is to get it in front of as many users as possible, so accessibility is crucial once we pass the MVP.

How GCP Can Support Our Research in the Future

Google Cloud Platform (GCP) can continue to support our research by

- Mentoring us. We are hungry for knowledge and eager to learn from Google's engineers.
- Improving the project directly. Solus is open-source so everyone can contribute to the codebase.
- Giving us access to Vertex AI's Generative Suite to test the capabilities of Google's language models. Once the language models are available for all Google Cloud users, we could add integrations so users can use Google Cloud's models with Solus.
- Supporting our organization. CSX Labs is an organization without a profit motive, so all monetary and equipment contributions are helpful. Examples of equipment contributions include Chromebooks. Examples of monetary contributions include money towards our incorporation, Office Space[1], and Google Cloud Credits towards our AI research.

Thank you for considering our proposal. We look forward to working with you in the future.

[1] See Bishop Ranch and Regus for more details on San Ramon office space.