AI-SmartContractGen Project Plan

Pang Bowen, Li Zhuokai, Wang Hongyu, Zhou Peiwen

September 30, 2023

1 Background

In finance, trade, manufacturing, logistics and many other areas, Web3 is presenting boundless new possibilities. The potential of Web3 and its blockchain technology is to fast-track innovation, inspire new business models, and make people's everyday lives better. The Hong Kong government has seen this and is committed to nurturing a vibrant Web3 ecosystem in Hong Kong in terms of both policies and resources. This ecosystem will become a significant driving force in Hong Kong's economic development. The manifestation of this trend is shown by the $50 million allocated by the Hong Kong government for Cyberport development and setting up a task force on promoting Web3 to spur ecosystem development in 2023 (*HK set to tap Web3 potential*, 2023).

This shows that society has an urgent need for the development of Web3. Being an indispensable constitution of Web3, there are higher requirements for blockchain technology. Furthermore, smart contracts, as code scripts running on the blockchain platform, have been applied to a large number of essential applications in the real world. Ordinary people also have higher requirements and greater needs for it.

Recognizing the necessity, we discerned a need for a tool capable of helping individuals test their smart contracts and give suggestions about how to revise the bugs detected. As the smart contract cannot be modified, it is very important to write the right ones that can meet the given conditions. Therefore, it can be believed that an

AI model trained with a large amount of high-quality data can become a reliable tool to detect possible problems in smart contracts inputted by users, which will greatly improve the efficiency of users in writing deployable smart contracts.

2 Objectives

The main objective of this project is to develop a model specialized in providing an auxiliary tool to locate potential bugs for Web3 developers. Our final deliverable will be a website or VS Code plugin integrating the model, enabling users to detect possible bugs in their codes. The product is required to implement the following following features.

As the deliverable of this project targets different levels of developers working on Solidity, our product allows user to post their code. Suggestions on potential bugs will be shown to the user as references, allowing users to make further adjustments conveniently.

2.1 Solidity Code Understanding

After receiving input from the user, the model used in this project should be capable of understanding the code written in Solidity, finding the probably vulnerable components in the program, and showing the problem to the user.

Our product should be capable of correctly locating the weaknesses of the code, which is the primary requirement, and make proper suggestions, based on the model's understanding, to fix the weaknesses if possible.

2.2 Secure and Error Detection

The immutability of smart contracts imposes substantial requirements on the securities of the contract code. It has been the case where contract fragility has caused the loss of millions of dollars. Without compulsory blockchain regulations and partially because of the anonymity of blockchain addresses, such losses will be unlikely to be traced and charged back. Hence, we hope the products of our model will help users' smart contracts minimize the possibility of being attacked.

The suggestion given by the model is expected to either focus on the logic issues or the safety of the input program, which are the main two problems that a programmer faces during the coding process. This functionality is believed to effectively save the time spent on testing the code, and significantly decrease the time from the beginning of the project to the deployment of the final smart contract.

2.3 Natural Language Generating

Instead of using the same model trained originally, an improvement will be added to this project, which will be achieved by improving the model to realise natural language generating and thus to output an explanation of the issue detected. By catching data to enrich the dataset automatically and periodically, followed by re-training the model to implement self-update, this feature will be added to the project finally.

As a promotion to the basic construction of the model, this objective will be realized after completing all the work that should be done to produce a workable model first, which may result in a lack of time for finishing this feature.

Date	Schedule
Oct 1	Phase 1 - Planning and Initial Setup • Completed Project Plan • Established Project Website
Oct 1- Oct 31	Phase 2 - Development and Preliminary Design • Training data crawling • Data cleaning • Primary model design
Nov 1- Nov 30	• First stage of training and model optimization • Model evaluation
Dec 1- Dec 31	Phase 3 - Enhancement and Mid-project Deliverable • Model Improvement • Develop associated terminal CLI as mid-project deliverable • Result visualization • Interim project report
Jan 1- Jan 31	• Website and model API design
Feb 1 - Apr 25	Phase 4 - Integration and Final Testing • Website development • Final Testing • Final Report

Table 1: Planned Schedule.

3 Methodology

3.1 Data Mining

As the beginning part of this project, data mining is considered one of the most significant components of the whole process, which directly affects the final performance of the model. To ensure the conciseness and safety of the Solidity code, famous programs running on blockchain for more than a period of time without showing any losses will be collected to constituent the dataset. During the process, python programs are needed to crawl the data, and there will be a threshold to justify whether the code is worth collecting.

The main venue of our data source will be GitHub and Etherscan. The range of the code selection includes but is not limited to popular ERC20 tokens and NFTs, basic token marketplace or AMM, and simple Wallet, which may cover most topics in DeFi and Web3. We may mainly acquire codes that have received more than 1000 transactions in a week to filter out codes with low quality.

3.2 Model Selection

The model used in this project will take an example by GPT, or more specifically, GPT-1. Several Transformers are expected to be displayed in the model to achieve both natural language understanding and smart contract generation work.

3.3 Model Evaluation

This project will adopt the Bilingual Evaluation Understudy(BLEU) score as one of the evaluation methods of the model. This score will be used to check the code generated by the model and the code written by humans. Another preference would be the ChrF metric, which evaluates the model to character base compared with word base. A more detailed scoring method is left to be discussed.

We may make use of the pass@k metric(Chen et al., 2021) to evaluate the model's ability to accomplish certain tasks. It will be possible to design new problems or modify existing problems to adapt to tasks specialized in Web3.

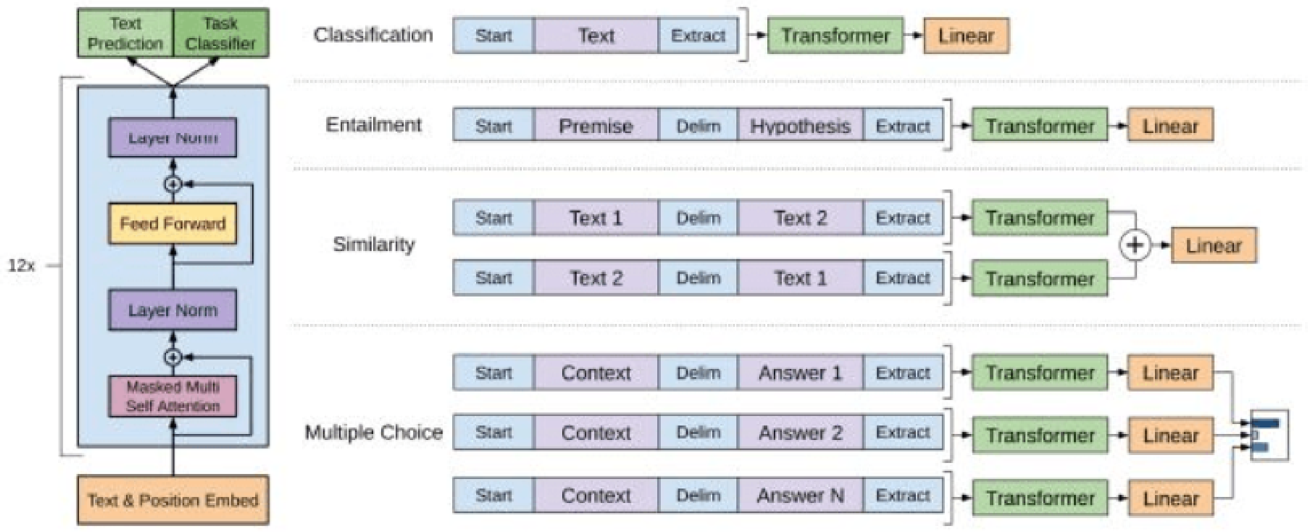

Figure 1: GPT Model Example

References

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., . . . others (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Hk set to tap web3 potential. (2023). Hong Kong's information Services Department. Retrieved from https://www.news.gov.hk/eng/2023/08/20230825/20230825_124937_227.html