DevGPT: Studying Developer-ChatGPT Conversations

Tao Xiao
Nara Institute of Science and Technology
Nara, Japan
tao.xiao.ts2@is.naist.jp

Christoph Treude
The University of Melbourne
Melbourne, Australia
christoph.treude@unimelb.edu.au

Hideaki Hata
Shinshu University
Nagano, Japan
hata@shinshu-u.ac.jp

Kenichi Matsumoto
Nara Institute of Science and Technology
Nara, Japan
matumoto@is.naist.jp

I. Title of data set

DevGPT

II. High-level overview

The emergence of large language models (LLMs) such as ChatGPT has disrupted the landscape of software development. Many studies are investigating the quality of responses generated by ChatGPT, the efficacy of various prompting techniques, and its comparative performance in programming contests, to name a few examples. Yet, we know very little about how ChatGPT is actually used by software developers. What questions do developers present to ChatGPT? What are the dynamics of these interactions? What is the backdrop against which these conversations are held, and how do the conversations feedback into the artifacts of their work? To close this gap, we introduce DevGPT, a curated dataset which encompasses 17,913 prompts and ChatGPT's responses including 11,751 code snippets, coupled with the corresponding software development artifacts—ranging from source code, commits, issues, pull requests, to discussions and Hacker News threads—to enable the analysis of the context and implications of these developer interactions with ChatGPT.

To create DevGPT, we leveraged a feature introduced by OpenAI in late May 2023, which allows users to share their interactions with ChatGPT through dedicated links.[1] We collected all such links shared on GitHub and Hacker News at six specific points in time: July 27, 2023, August 3, 2023, August 10, 2023, August 17, 2023, August 24, 2023, and August 31, 2023. If users chose to delete or deactivate their shared conversations in the intervening periods, we ensured data consistency by accessing the original shared link across all these snapshots.

Table I provides an overview of the snapshot 20230831. Comprising 2,891 shared ChatGPT links sourced from 2,237 GitHub or Hacker News references, the dataset contains a total of 17,913 prompts/answers. This includes 11,751 code snippets, with Python (1,735), JavaScript (1,530), and Bash (1,435) as the top three programming languages. 546 of these links are referenced across multiple sources, resulting in a unique count of 2,345 individual ChatGPT shared links within DevGPT. We will periodically expand the DevGPT dataset until its official release for the MSR mining challenge. Figure 1 shows an instance of a ChatGPT conversation from the dataset, together with the pull request it was related to and how the code was updated after the ChatGPT conversation.

[1]https://help.openai.com/en/articles/7925741-chatgpt-shared-links-faq

III. Internal structure

The dataset consists of a collection of JSON files collected from the six sources detailed in Table I. For each source, we provide distinct metadata in the JSON file to enable source-specific analysis. Apart from the source-specific metadata, every JSON contains a consistent attribute: a list of shared ChatGPT links. Each shared link includes the URL to the ChatGPT conversation, the associated HTTP response status codes, the access date of the URL, and the content within the HTML response. Additionally, each conversation contains a list of prompts/answers, inclusive of any code snippets. We provide details including the date of the conversation, the count of prompts/answers, their token information, and the model version involved in the chat. Attributes detailing where the conversation was referenced are also included—such as the referencing URL, the nature of the mention (e.g., a comment), the individual who mentioned it, and the context in which it was cited. A comprehensive breakdown of the data structure is available at https://github.com/NAIST-SE/DevGPT. Additionally, we provide a CSV file cataloging all shared ChatGPT links gathered from GitHub and Hacker News.

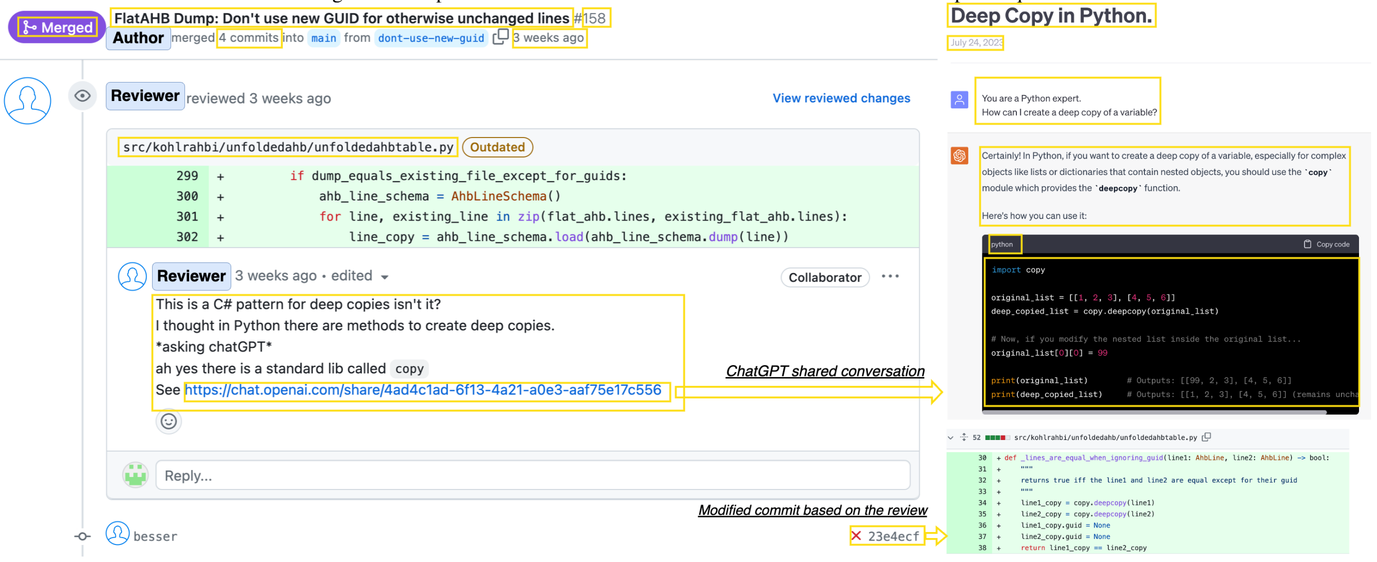
IV. How to access

The DevGPT dataset is available for download on Zenodo, see Section VI. It is formatted in JSON, making it easily parsable with any standard JSON library. Additionally, we include the HTTP response, which can be analyzed using any HTML parser. The dataset also categorizes code snippets by

TABLE I
SUMMARY STATISTICS OF THE SNAPSHOT 20230831

Sources	#	Mentioned in	Shared ChatGPT Links			ChatGPT Conversations	
			# Shared Links	# Accessible Links	# Conversations with Code	# Prompts	# Code Snippets
GitHub Code File	970	Code	1386	1306	577	12320	7190
GitHub Commit	481	Message	481	477	470	1387	1569
GitHub Issue	353	Comment	264	243	160	844	739
		Description	150	138	98	854	757
		Title	3	3	3	49	77
Hacker News	195	Comment	274	240	50	837	163
		Attached URL	44	39	3	380	65
		Story	17	14	4	57	66
GitHub Pull Request	193	Description	82	81	57	494	428
		Review Thread	74	67	48	154	146
		Comment	65	60	35	351	397
GitHub Discussion	45	Comment	25	22	11	72	37
		Description	20	19	12	89	94
		Reply	6	5	3	25	23

Fig. 1. Example of a ChatGPT conversation in the context of a GitHub pull request


Merged
FlatAHB Dump: Don't use new GUID for otherwise unchanged lines #158
Author merged 4 commits into main from dont-use-new-guid 3 weeks ago
Reviewer reviewed 3 weeks ago
View reviewed changes
src/kohlrahbi/unfoldedahb/unfoldedahbtable.py Outdated

```
+         if dump_equals_existing_file_except_for_guids:
+             ahb_line_schema = AhbLineSchema()
+             for line, existing_line in zip(flat_ahb.lines, existing_flat_ahb.lines):
+                 line_copy = ahb_line_schema.load(ahb_line_schema.dump(line))
```

Reviewer 3 weeks ago • edited
Collaborator

This is a C# pattern for deep copies isn't it?
I thought in Python there are methods to create deep copies.
asking chatGPT
ah yes there is a standard lib called copy
See https://chat.openai.com/share/4ad4c1ad-6f13-4a21-a0e3-aaf75e17c556

ChatGPT shared conversation
Reply...
Modified commit based on the review
besser
23e4ecf
Deep Copy in Python.
July 24, 2023

You are a Python expert.
How can I create a deep copy of a variable?

Certainly! In Python, if you want to create a deep copy of a variable, especially for complex objects like lists or dictionaries that contain nested objects, you should use the `copy` module which provides the `deepcopy` function.

Here's how you can use it:

```python
import copy

original_list = [[1, 2, 3], [4, 5, 6]]
deep_copied_list = copy.deepcopy(original_list)

# Now, if you modify the nested list inside the original list...
original_list[0][0] = 99

print(original_list)        # Outputs: [[99, 2, 3], [4, 5, 6]]
print(deep_copied_list)     # Outputs: [[1, 2, 3], [4, 5, 6]] (remains uncha
```

```
+ def _lines_are_equal_when_ignoring_guid(line1: AhbLine, line2: AhbLine) -> bool:
+     """
+     returns true iff the line1 and line2 are equal except for their guid
+     """
+     line1_copy = copy.deepcopy(line1)
+     line2_copy = copy.deepcopy(line2)
+     line1_copy.guid = None
+     line2_copy.guid = None
+     return line1_copy == line2_copy
```

type, enabling researchers to use corresponding compilers for execution. No credentials are needed to access the dataset.

V. POTENTIAL RESEARCH QUESTIONS

The following provides a sample list of research questions that can be answered with the DevGPT dataset:

1) What types of issues (bugs, feature requests, theoretical questions, etc.) do developers most commonly present to ChatGPT?
2) Can we identify patterns in the prompts developers use when interacting with ChatGPT, and do these patterns correlate with the success of issue resolution?
3) What is the typical structure of conversations between developers and ChatGPT? How many turns does it take on average to reach a conclusion?
4) In instances where developers have incorporated the code provided by ChatGPT into their projects, to what extent do they modify this code prior to use, and what are the common types of modifications made?
5) How does the code generated by ChatGPT for a given query compare to code that could be found for the same query on the internet (e.g., on Stack Overflow)?
6) What types of quality issues (for example, as identified by linters) are common in the code generated by ChatGPT?
7) How accurately can we predict the length of a conversation with ChatGPT based on the initial prompt and context provided?
8) Can we reliably predict whether a developer's issue will be resolved based on the initial conversation with ChatGPT?
9) If developers were to rerun their prompts with ChatGPT now and/or with different settings, would they obtain the same results?

VI. LINKS

https://github.com/NAIST-SE/DevGPT and https://doi.org/10.5281/zenodo.8304091