

README: Reproduce Results

for “Generative AI for Economic Research: Use Cases and Implications for Economists”
by Anton Korinek, 2023, Journal of Economic Literature

This repository contains the data and python source code for processing and obtaining completions for the list of chat prompts from the OpenAI API included in the paper. You will need to ensure that you have installed python with the correct packages as outlined below, and then follow the steps in section **Usage** below. The output will be two files with the completions, one in plain text/markdown format and one in csv format with the relevant metadata. Moreover, this README file also contains instructions for how to access ChatGPT Advanced Data Analysis and Claude 2 to obtain chat completions.

Data and Code Availability Statement

The chat prompts that generated the results presented in the paper by Korinek (2023) are contained in the file `prompts.xlsx` in this repository. The completions obtained using OpenAI GPT-3.5 and GPT-4 can be obtained using the OpenAI API via the python program `generate.py` that is contained in this repository. A sample of the output that is generated is given in the repository in the files `results.csv` and `results.txt`. Replicators should expect slight variation in the responses, even with "temperature" set to "0". Responses from ChatGPT Advanced Data Analysis and Claude 2 with file uploads could not be generated by this code programmatically as of Sept. 2023 and needed to be entered manually.

Computational Requirements

Software: I employed the following software and package versions to produce the results:

- To execute the file `generate.py`, I employed Python 3.11.3 with packages
 - `numpy 1.24.3`
 - `pandas 2.0.3`
 - `openai 0.27.10`

Memory and Runtime Requirements: The computational requirements on the user side are minimal. A simple laptop with an up-to-date Python installation and the OpenAI package is sufficient. However, the code queries OpenAI's servers via an API, where a significant amount of computation is performed to complete chat prompts. This requires setting up an OpenAI account and payment method (see below). The reproduction takes around 10 minutes to run.


Setting Up an OpenAI Account for GPT-3.5/GPT-4 Completions

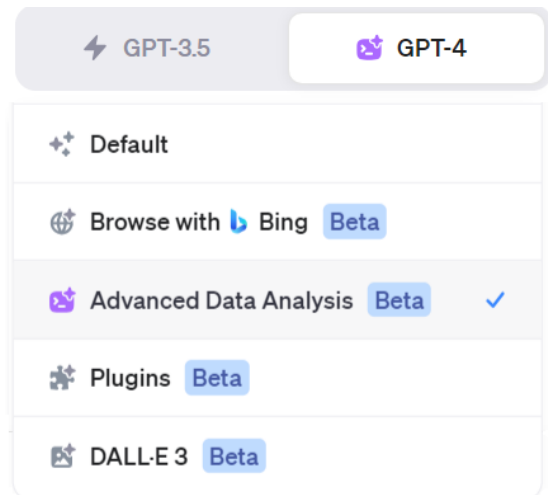
Executing the provided code requires an OpenAI account and associated API key. To set up an OpenAI account, use the following link: <https://platform.openai.com/>. Click “Sign up” at the top

right to create an account. In order to access the API, you will need to attach a payment method to the account. Before accessing the API, be sure to note the associated cost from <https://openai.com/pricing>. As of September 2023, executing the reproduction code cost around 50 US cents.


Once your account is set up, go to “View API keys” and create a new key. You can either set this key as an environment variable `OPENAI_API_KEY` in your operating system, or manually enter the key in the python program, or enter it whenever prompted by the program.

Setting Up an Account to Use ChatGPT Advanced Data Analysis

Using the Advanced Data Analysis tool in ChatGPT requires signing up for the Plus version of ChatGPT at <https://chat.openai.com>. Click “Sign up” at the right window to create an account and subscribe to the paid version of ChatGPT Plus, which automatically includes Advanced Data Analysis. As of September 2023, a subscription costs \$20/month. Once logged in, the user can hover over the “GPT-4” tab (see figure on the right) and choose the “Advanced Data Analysis” tool from the list of models. The user queries can then be entered in the chat box that appears. Files can be uploaded by clicking on the -button in the chat box. Chats 17 and 21 in the paper were produced by entering the described prompts in ChatGPT Advanced Data Analysis.



Setting Up a Claude 2 Account

A Claude 2 account can be created by visiting <https://claude.ai> and following the sign-up instructions. Claude 2 offers both a free version and a \$20/month Pro version that allows for more and faster usage. I used the free version. After registering, user queries can be entered in the chat box that appears. Files can be uploaded by clicking on the paperclip button  in the chat box. Chats 3, 11 and A.1 in the paper were produced by entering the described prompts in Claude 2.

Files in Replication Package

simple_example_chat1.py

- A simple python script to get started and generate the completion in Chat 1.
- Ensure you have set the correct OpenAI API key or enter it during execution.

prompts.xlsx

- An Excel file containing all chat prompts and associated metadata.
- Columns:

- Chat: Identifier for the chat session.
- Category: Category number for the chat.
- Lab: AI lab associated with the chat.
- Model: Model used for the chat. (e.g., gpt-4-0613 or text-davinci-003)
- Prompt: Text of the actual chat prompt.

generate.py

- Python script to process chat prompts from *prompts.xlsx* and obtain completions using the OpenAI API.
- Uses OpenAI's ChatCompletion and Completion API endpoints based on the model type specified in the Excel file.
- Ensure you have set the correct OpenAI API key or enter it during execution.
- Additionally, if you would like to only reproduce a certain list of prompt completions rather than all of them, change line 39.
- The completions are stored in *results.csv* and *results.txt*.
- *Note:* Occasionally, due to variability in traffic received by OpenAI, you may get request timeout errors. This will not always occur and thus simply rerunning the code should fix it.

Usage

Ensure you have the necessary Python libraries installed:

```
pip install openai pandas numpy
```

To execute the simplest example, run the script:

```
python simple_example_chat1.py
```

The results will be displayed on the screen.

To execute the full replication code, run the script:

```
python generate.py
```

Open the generated files *results.csv* and/or *results.txt* to view the results.

Note that the results differ from execution to execution and are therefore not perfectly reproducible, even if the temperature parameter is set to zero to minimize variability. The reasons are discussed in the part “Reproducibility” in Section 3.1 of Korinek (2023).

To repeatedly execute the code without having to manually enter the OpenAI API every time, you may find it convenient to save your OpenAI API key as a system variable under the name `OPENAI_API_KEY`. Under Microsoft Windows, this can be done by

opening a PowerShell session and entering:

```
[Environment]::SetEnvironmentVariable("OPENAI_API_KEY",  
    "your_api_key_here", "Machine")
```

ChatGPT can offer help with how to set this system variable in other operating systems. Alternatively, if you do not set a system variable, the script will simply prompt you to enter your OpenAI API key manually. You could also enter the API key directly in line 34 of the python code, but this risks compromising the security of your key if you share the file with other users.

Outputs

results.csv contains the original chat prompts and metadata along with the obtained completions and token usage details.

results.txt shows the chat sessions with prompts and completions in text format.

Acknowledgment

We thank Lars Vilhuber for his thoughtful guidance and advice in creating this replication package.

License



This README was produced by Anton Korinek and Davis Taliaferro in 2023 and is distributed under a CC-BY-NC license.