README by Echo Mulder, Gray Lewis and Nick Lodder

https://github.com/graylewis/GPT-AA

GPT-AA (GPT-3 Academic Assistant)

Project contributors: Echo Mulder, Gray Lewis, and Nick Lodder

GPT-AA is a set of tools for interacting with the GPT-3 API for the purpose of human-free open question grading, and aims to provide a greatly simplified interface for semi-technical/hobbyist users with little to no programming experience. GPT-AA constitutes a constraint & abstraction layer for interacting with GPT-3 as a few-shot classifier, as opposed to its classical application as a purely generative AI.

Usage of the tool does NOT eliminate human bias from the grading process since the training process relies on human-generated input both for the formation of the implicit rubric and the pre-training of the GPT-3 model. This can be partially alleviated by strictly following an existing rubric when creating the few-shot training examples.

Setup

GPT-AA uses pipenv for managing project dependencies. To install pipenv, run the following command in pip:

```
pip install pipenv
```

Subsequently, the dependecies required to run GPT-AA can be installed using the following command:

```
pipenv install
```

And finally each python script can then be executed inside the Pipenv shell. The pipenv shell can be accessed with `pipenv shell` and then the scripts can be executed normally, or an individual program can be run with

```
pipenv run python script.py
```

(While this process may seem unnecessarily complicated, it is important to avoid cluttering globally installed dependencies)

Providing authentication for GPT-3's API is as simple as creating a .env file and specifying the API key within that file as follows:

```
api_key="abcdefghijklmnopqrstuvwxyz"
```

You're good to go! The next section outlines the role of each of the python scripts in interacting with GPT-AA.

Core Concepts

1. Examples

- "Given any text prompt, the API will return a text completion, attempting to match the pattern you gave it. You can "program" it by showing it just a few examples of what you'd like it to do; its success generally varies depending on how complex the task is. The API also allows you to hone performance on specific tasks by training on a dataset (small or large) of examples you provide, or by learning from human feedback provided by users or labelers."
- GPT-3 is a general AI, meaning that it is capable of a wide variety of tasks. In our case, we are looking to use GPT-3 as a classification algorithm.
- Using GPT-3 is only permitted through integration with its API. Since GPT-3 can parse english, feeding it a handful of examples is sufficient information for GPT-3 to infer deep levels of context and meaning from both the labels and the examples and form an implicit rubric based on the qualities of the examples and their relation to the labels.
- This means, for example, that GPT-3 can interpolate between scores i.e. GPT-3 could be fed examples of a 0/10, 5/10, and a 10/10 and be able to infer what a 3/10 or 7/10 would look like. Additionally, it means that GPT-3 can understand and hence grade for more ephemeral qualities like succinctness and clarity.

2. The Config File

- The examples being provided in the form of a JSON file form an implicit rubric which GPT-3 uses to grade novel answers. It's important to note that usage of the tool does NOT eliminate human bias from the grading process since the training process relies on human-generated input both for the formation of the implicit rubric and the pre-training of the GPT-3 model.
- genConfig.py is a convenience tool for inputting these training examples into the required format for classification. Essentially, genConfig.py will ask for a series of answers at specific scores, and then store them in an appropriately formatted JSON file. Any (sensible) novel answer can be scored against a generated configuration file. In order to simplify the process, genConfig.py always asks for 5 answers at specified scores. While this is generally sufficient to get decent grading, high quality grading is

much more likely when creating a dataset based off of real answers and real corresponding grades.

- Manual creation of a config file is relatively straightforward, and can improve accuracy by specifying more examples where needed. The expected format is a JSONLINES (https://jsonlines.readthedocs.io/en/latest/) file where each line is a self-contained JSON-compliant object with a field called "text" and a field called "label", where the text field is the answer and the label is the associated score.
  - e.g. `{"text": "Data types in python include, blek, smhleck, and zek.", "label": "0/10"}`

3. GPT-3 Remote File Upload

- In order to maximize accuracy, GPT-AA uses GPT-3's most advanced classification model, which requires the examples to be stored remotely on the OpenAI server. The file outputted by genConfig.py is formatted in line with OpenAI's standard.
- uploadFile.py is a convenience tool for uploading the output of genConfig.py (by default saved to "gpt3Config.json"), or any JSONLINES file that conforms to the standard outlined above. After uploading, the program will return the metadata necessary to point to the remotely hosted file. The "id" field of the metadata output will ultimately be used as an input into the querying program. IMPORTANT!: After uploading the file, uploadFile.py will return a file id that will need to be recorded for use in the primary script. uploadFile.py supports 1 command line argument:"
  - -f="filePath" or –file="filePath" This argument provides a custom path for the file being uploaded. If not specified, uploadFile.py will assume the configuration file is located at "./gpt3Config.json", the output file name for genConfig.py.

4. Queries

- GPT-3 refers to inputs to the AI as "queries". In this case, our query is the answer we want to be graded based on the implicit rubric created by the configuration file.
- After uploading a configuration file and saving its corresponding id, the main program is ready to be used.
- Index.py is the entry point for the primary functionality. Index.py accepts 2 primary command line arguments:
  - -f="fileId" or –file="fileId" This argument provides a file ID to be referenced by the classification function, and corresponds the the output of uploadFile.py. e.g. >> python index.py –file="file-ksfjdskajfkjKXJKF"

  - -q="Answer to be graded" or –query="Answer to be graded" This argument provides an answer to be graded against the configuration file's examples. e.g. >> python index.py –query="This is the answer to be graded"

- Index.py also defaults to prompts when command lines arguments are not provided, although arguments are the preferred way to input the parameters.

Technical Advancements

This section will aim to clarify which pieces of GPT-AA can be considered advancements in terms of interacting with GPT-3.

1. Testing - GPT-AA is the result of hundreds of iterations of formats and approaches that ultimately taught us valuable lessons about which situations enable GPT-3 to operate most effectively as a classifier. For example, several approaches to actually inputting and outputting examples to GPT-3 were used before the on-site classification set approach was settled upon.
   - Other criterion we identified as important include:
   - The range of labels (scores) being considered; A scale of 1-10 strikes a balance between flexibility and introduction of noise through the consideration of too many potential labels.
   - The inclusion/non-inclusion of the question being answered; while GPT-3 demonstrates impressive results WITHOUT examples (only based on the question), we found that including the question when examples were considered tended to lead to less accurate results.
2. Ease of Use & Flexibility - While most applications built with the GPT-3 API are focused on creating tools for completely non-technical people, we wanted to build a certain level of flexibility into the tool while still providing intuitive tools for bridging the gap between technical and non-technical users. An example of this principle in action is the intentional decision to avoid the use of GUIs, and to build auxiliary utilities to provide a more intuitive interface where needed. This flexibility allows the user to define their configuration (implicit rubric/examples) exactly as they would like and maximizes potential accuracy.
3. Novel Application - No project with a similar concept at any level of functionality was able to be found by our team. We believe that tools similar to ours are inevitably going to become popular in the future.