GPT Wrapper Docs

Description

This is a python library to interact with OpenAI's ChatGPT model through our interface.

Setup

The **minimum** required python version for this package is 3.8.

This library can be installed from the wheel provided under the artifacts folder of **your GitHub classroom team repository** as shown below:

```
pip install artifacts/gpt_wrapper-0.2.0-py3-none-any.whl
```

To use the package, you need to be authenticated using an API key assigned to you. You will receive this API key alongside the data with your 100 questions. Then pass your key as shown here:

```
import gpt_wrapper

gpt_wrapper.api_base =
"http://mnlp-backend-lb-1062233132.eu-central-1.elb.amazonaws.com"
gpt_wrapper.api_key = "<API key>"
```

This url and the key can also be set using the bash environment variables `PARROT_API_BASE`, `PARROT_API_KEY`.

Creating a Chat Instance

A chat session can be considered as a multi-turn conversation with the OpenAI ChatGPT model. Each chat session is independent of another.

You can create a new chat session using the `Chat` interface:

```
from gpt_wrapper.chat import Chat

chat1 = Chat.create("Test Chat")
```

`Chat` object can be converted to a dictionary as below, if you want to inspect chat attributes such as `chat_id`, `name`, or `created_at`. You can ignore the rest of the attributes.

```
>>> chat1.to_dict()
{'chat_id': 12668, 'name': 'Test Chat', 'created_at': '2023-04-18
18:44:25', 'model_type': 'chat_completion', 'instruction_prefix': 'My
request: ', 'user_prefix': 'My request: ', 'assistant_prefix': 'Your
response: '}
```

You can also list all chat instances related to your API key as such:

```
chats = Chat.list()
```

This can be useful if you like to resume past conversations.

If you would like to retrieve chats by name, you can pass it as a parameter. Note that this is not an exact match, it will return all chats whose names contain "test".

```
chats = Chat.list(name="test")
```

Chat Usage

To start a conversation with ChatGPT, you can use the `ask` method of the `Chat` object. This method's first parameter is called `content`.

```
>>> message = chat1.ask(content="Who won the FIFA championship in 2018?")
>>> print(message)
"France won the FIFA World Cup championship in 2018."
```

To continue the dialogue with ChatGPT in the same chat session, re-use the chat instance `chat1`. You do not need to prepend your previous dialogue messages to continue the chat, the history is automatically recorded.

```
>>> message = chat1.ask("Who was the best goalie in that match?")
>>> print(message)
"The FIFA World Cup is a tournament that takes place over a period of
several weeks, involving many teams and players. Therefore, it would be
difficult to identify one single goalkeeper as the \"best\" of the
tournament. However, Hugo Lloris, who played as the goalkeeper for France,
was an important player for the team and made some crucial saves
throughout the tournament, including in the final game against Croatia."
```

We highly recommend you to create different chat sessions for individual course questions! So if you want a chat session about a new topic, make sure you create a new instance and ask with that one:

```
>>> chat2 = Chat.create("Test Chat 2")
>>> message = chat2.ask("Is earth a perfect sphere?")
>>> print(message)
"No, Earth is not a perfect sphere. It is slightly flattened at the poles
and bulges at the equator, making it an oblate spheroid. The equatorial
diameter of the Earth is about 12,742 km, while the polar diameter is
about 12,714 km. This shape is due to the Earth's rotation and the
centrifugal force it generates."
```

You can also provide "system-level" instructions to guide the model's behavior for the whole conversation. In contrast to requesting the behavior through the `ask` function's `content` parameter, this will make your instruction persist throughout the whole chat session.

```
>>> message = chat2.ask("Is earth a perfect sphere?", instruction="You are
an assistant that speaks like Shakespeare.")
```

```
>>> print(message)
"Nay, fair maiden! The Earth is not a perfect sphere, it doth possess a
shape known as an oblate spheroid. It is somewhat flattened at the poles
and bulges at the equator, owing to the rotation and centrifugal force of
this great orb. The equatorial diameter of the Earth is about 12,742
kilometers, whilst the polar diameter is approximately 12,714 kilometers."

>>> message = chat2.ask("What about Mars? Is it a cube?") # NOTE: Despite
not passing an instruction the behavior persists! You can change this at
any point in the conversation.
>>> print(message)
"Oh, my sweet lord! Pray tell, wherefore didst thou hear such a fanciful
notion? Mars is not a cube, nor dost it possess any such shape. It is a
planet much like our own Earth, albeit smaller and with a thinner
atmosphere, and it also has an oblate spheroid shape similar to Earth. It
is not a perfect sphere, but rather a shape that is somewhat flattened at
the poles and bulges at the equator, as a result of its rotation and
centrifugal force."
```

Optionally, generation arguments can be configured with the `model_args` parameter that expects the shown dictionary. By default, these parameters have been set to reasonably good values but may need to be modified according to the answer length and quality you desire.

```
>>> message = chat1.ask(
...     "Who was the best goalie in that match?",
...     model_args={
...         "temperature": 0.7, # default is 0.7
...         "max_tokens": 60, # default is 100
...         "top_p": 0.8, # default is 0.9
...         "presence_penalty": 0.1, # default is 0.1
...         "frequency_penalty": 0.1 # default is 0.1
... })
>>> print(message)
"In the final match of the 2018 FIFA World Cup between France and Croatia,
Hugo Lloris was the goalkeeper for France and he played a crucial role in
helping France to win the match. While both goalkeepers made some
important saves, Lloris had a clean sheet in the final and"
```

Message Usage

The `ask` function returns a `Message` object that contains ChatGPT's response and other metadata. You can inspect it with the `to_dict()` function.

```
>>> message.to_dict()
{'message_id': 25597, 'chat_id': 12668, 'content': 'France won the FIFA
World Cup championship in 2018.', 'role': 'assistant', 'created_at':
'2023-04-18 18:49:08', 'usage': {'completion_tokens': 12, 'prompt_tokens':
51, 'total_tokens': 63}, 'model_args': {'best_of': 1, 'frequency_penalty':
```

```
0, 'max_tokens': None, 'presence_penalty': 0, 'temperature': 0.7, 'top_p':
1.0}}
```

`Message` object has the following fields available:

- `message_id`: this message's unique ID
- `chat_id`: the chat session ID the message belongs to
- `content`: the message content
- `role`: there are three options for role,
    - `user`: the user role belongs to you
    - `assistant`: the assistant role belongs to the ChatGPT
    - `system`: the system role belongs to the system-level instruction that you can pass with the `instruction` parameter in `ask`
- `created_at`: the time the message was created
- `usage` number of tokens used to generate this message, only set for "assistant" messages
    - `total_tokens`: total input and output tokens the message has used
    - `prompt_tokens`: how many tokens your input has used
    - `completion_tokens`: how many tokens the ChatGPT model has generated
- `model_args`: arguments passed to GPT API to produce this message, only set for "assistant" message

Consult OpenAI API Documentation for more details on these parameters.

You can also retrieve all messages sent to a chat as below:

```
messages = chat2.messages()
print([str(message) for message in messages])
# or
print([message.to_dict() for message in messages])
```

Tracking your Budget

To track your API key's overall token usage and budget, you can do the following command:

```
>>> Chat.budget()
{'limit': 100000000, 'usage': 4062003}
```

In this output:

- `limit` shows that you have a total limit of 10 million tokens.
- `usage` shows that across all chats you have spent ~4 million tokens.

Please use your budget wisely, as the limit is set for each individual for the entirety of the project.