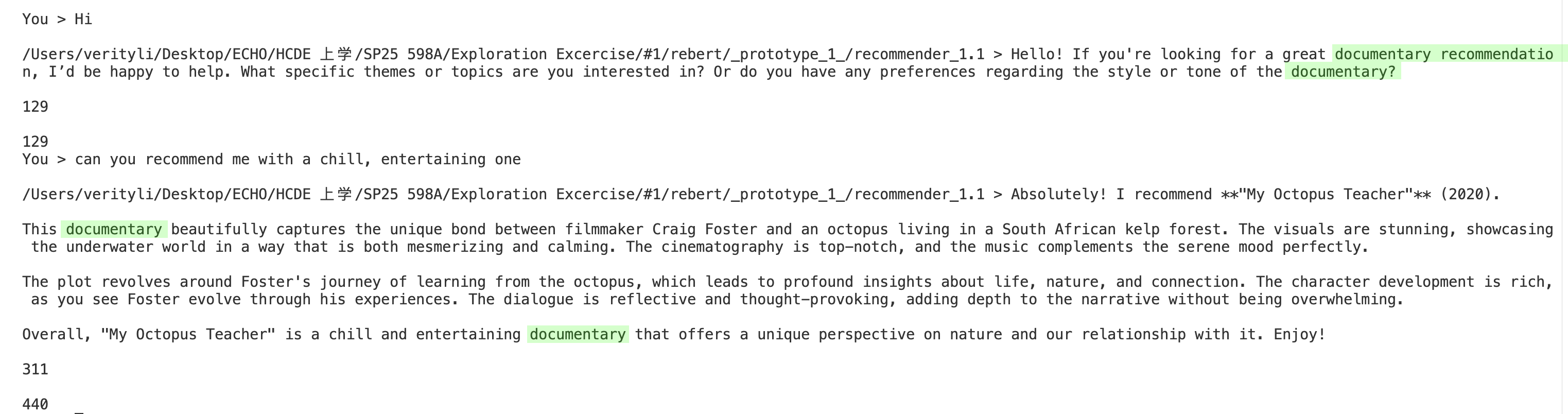
**Explore 1.1**

In prototype p1.v1 the code introduced a system message. That system message was used to create a movie recommender persona. The persona can be modified using existing prompting techniques. Try modifying the prompt to focus movie recommendations on a specific movie genre, such as action movies, romance movies, musicals, rom-com, sci-fi, or documentaries.

**Evidence 1.1**

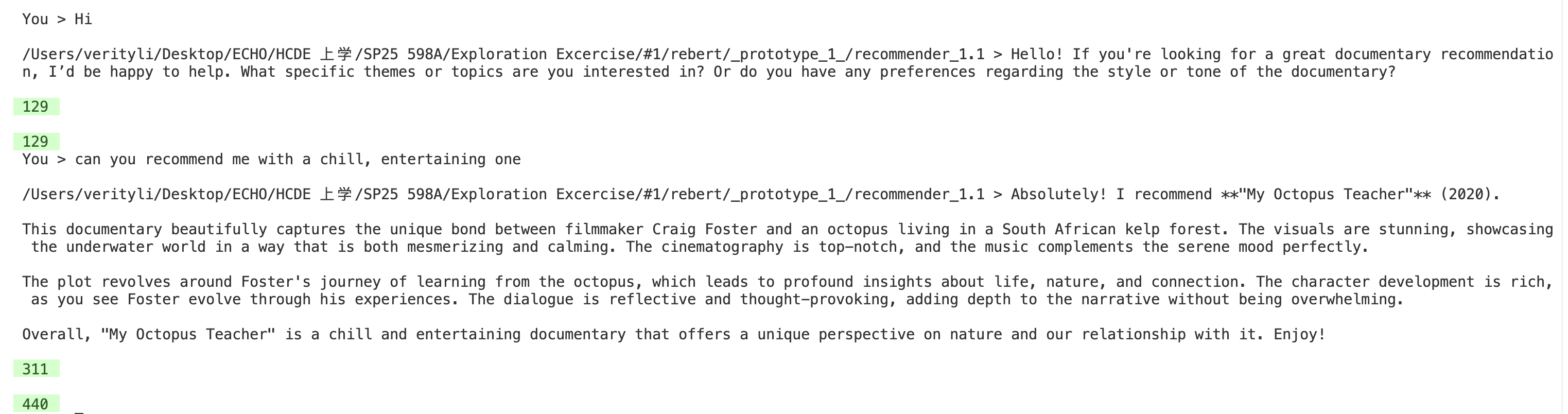
****

I decided to modify the prompt to make the recommendation system focus on documentaries. After updating the prompt message, we can see that the system now only provides documentary recommendations.

**Explore 1.2**

Many LLM services will have some way to cover their costs. These services are often charged through some form of usage. That is, users of an API are charged by the amount of text produced. The response that we receive in the function make\_chat\_request() includes the amount of usage. Modify make\_chat\_request() to return the usage for each request. Modify the loop in main() to sum the total usage for a chat session. Print the total usage when the session is completed.

**Evidence 1.2**

****

I modified the code to display the API usage for each request on the first line, and the total API usage on the second line.

**Explore 1.3**

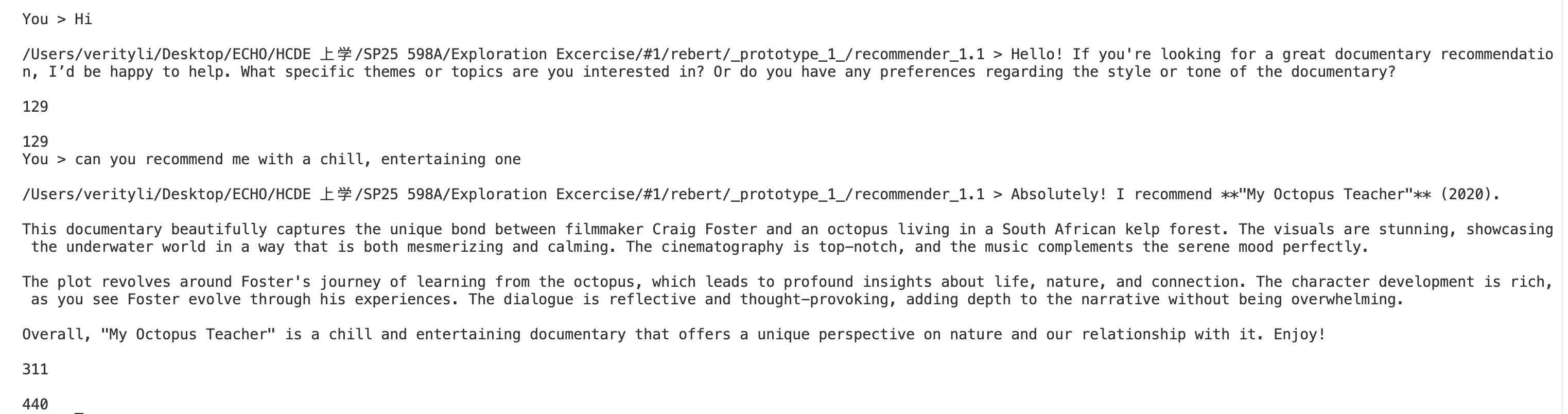
The API specifi cation allows us to send parameters in the request body that will change how the LLM model generates text. These parameters would become part of the chat\_context that we send in the request. Two valuable parameters to explore are temperature and max\_completion\_tokens. Modify the code to include two new constants called OAI\_MODEL\_TEMPERATURE and OAI\_MODEL\_MAX\_TOKENS. Then modify the new\_chat\_context() function to include the temperature and max\_completion\_tokens parameters and set them to the respective constants. Review the OpenAPI documentation to understand how responses change as you change the temperature and max\_completion\_tokens parameters.

**Evidence 1.3**

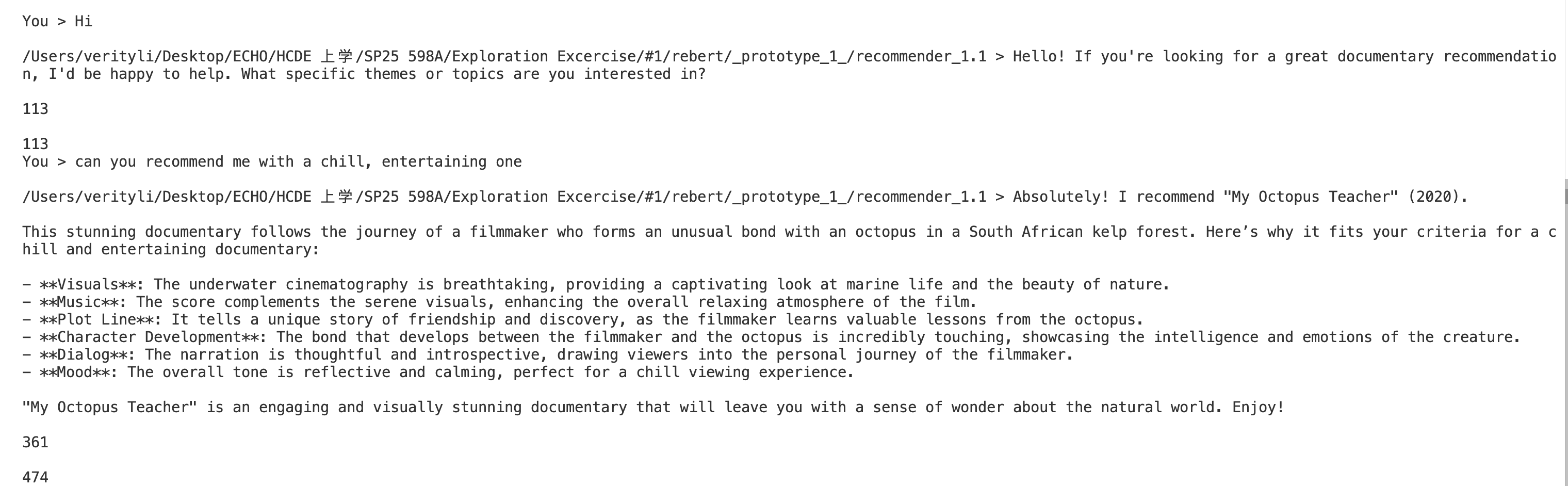
I read through the documentation and learned that the temperature controls how deterministic or creative the model’s responses will be. A lower temperature value results in more certain and logical answers, often leading to similar responses for the same question. On the other hand, a higher temperature will produce more creative and varied outputs.

As for the max tokens, it defines the maximum number of tokens the model can use to generate a response.

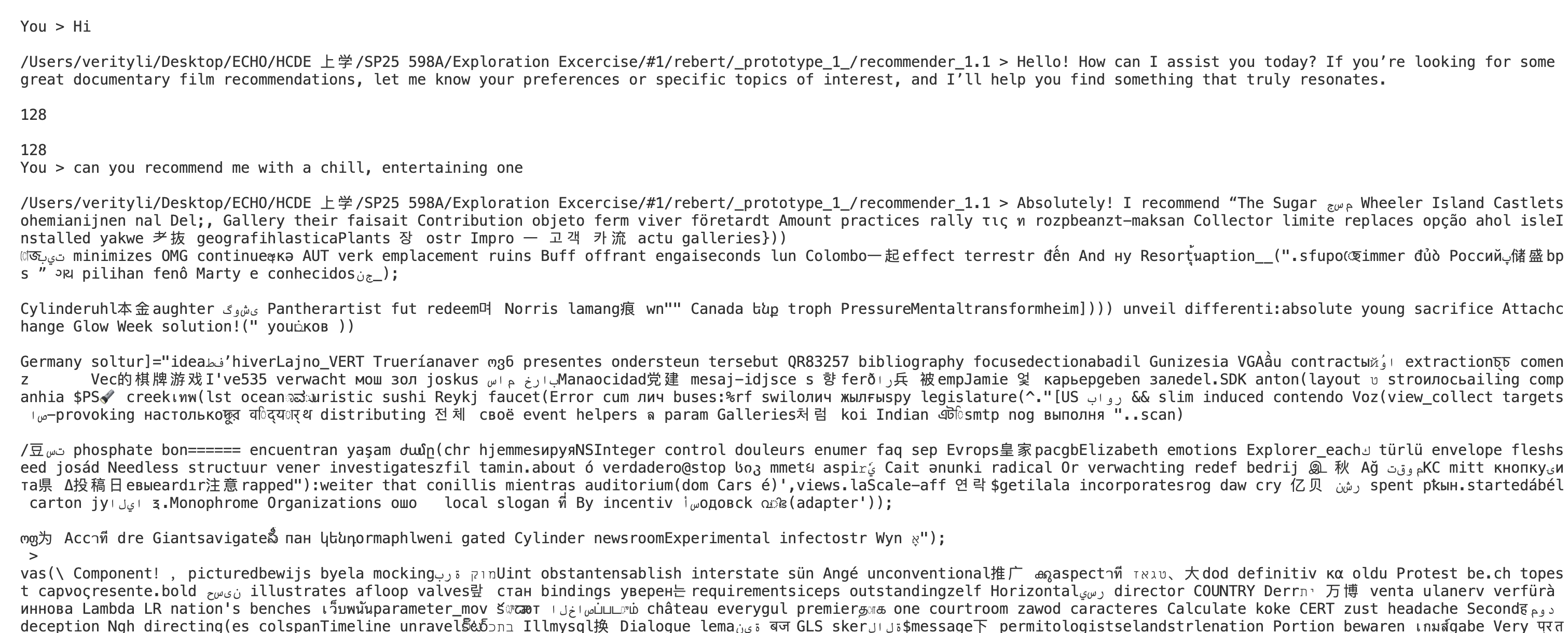
In the first scenario, I set the model temperature to 0, and the max tokens to 1000. This is the result:



In the second scenario, I set the model temperature to 1.5, and the max tokens to 1500. This is the result when asking the same question:



In the third scenario, I set the model temperature to 2, and the max tokens to 1500, and asked the same questions:



From the results, we can see that as the temperature and max tokens increase, the responses become longer and more elaborate. However, when the temperature gets too high, the randomness starts to impact readability and might result in outputs that are completely nonsense and unrecognizable.