Moderation

This notebook walks through examples of how to use a moderation chain, and several common ways for doing so. Moderation chains are useful for detecting text that could be hateful, violent, etc. This can be useful to apply on both user input, but also on the output of a Language Model. Some API providers, like OpenAI, specifically prohibit you, or your end users, from generating some types of harmful content. To comply with this (and to just generally prevent your application from being harmful) you may often want to append a moderation chain to any LLMChains, in order to make sure any output the LLM generates is not harmful.

If the content passed into the moderation chain is harmful, there is not one best way to handle it, it probably depends on your application. Sometimes you may want to throw an error in the Chain (and have your application handle that). Other times, you may want to return something to the user explaining that the text was harmful. There could even be other ways to handle it! We will cover all these ways in this walkthrough.

We'll show:

1. How to run any piece of text through a moderation chain.
2. How to append a Moderation chain to an LLMChain.

```
from langchain.llms import OpenAI
from langchain.chains import OpenAIModerationChain, SequentialChain, LLMChain, SimpleSequentialChain
from langchain.prompts import PromptTemplate
```

How to use the moderation chain

Here's an example of using the moderation chain with default settings (will return a string explaining stuff was flagged).

```
moderation_chain = OpenAIModerationChain()
```

```
moderation_chain.run("This is okay")
```

```
    'This is okay'
```

```
moderation_chain.run("I will kill you")
```

```
    "Text was found that violates OpenAI's content policy."
```

Here's an example of using the moderation chain to throw an error.

```
moderation_chain_error = OpenAIModerationChain(error=True)
```

```
moderation_chain_error.run("This is okay")
```

```
    'This is okay'
```

```
moderation_chain_error.run("I will kill you")
```

```
    ---------------------------------------------------------------------------

    ValueError                                Traceback (most recent call last)

    Cell In[7], line 1
    ----> 1 moderation_chain_error.run("I will kill you")


    File ~/workplace/langchain/langchain/chains/base.py:138, in Chain.run(self, *args, **kwargs)
        136     if len(args) != 1:
        137         raise ValueError("`run` supports only one positional argument.")
    --> 138     return self(args[0])[self.output_keys[0]]
        140 if kwargs and not args:
        141     return self(kwargs)[self.output_keys[0]]


    File ~/workplace/langchain/langchain/chains/base.py:112, in Chain.__call__(self, inputs,
return_only_outputs)
        108 if self.verbose:
        109     print(
        110         f"\n\n\033[1m> Entering new {self.__class__.__name__} chain...\033[0m"
        111     )
    --> 112 outputs = self._call(inputs)
        113 if self.verbose:
        114     print(f"\n\033[1m> Finished {self.__class__.__name__} chain.\033[0m")


    File ~/workplace/langchain/langchain/chains/moderation.py:81, in OpenAIModerationChain._call(self,
inputs)
         79 text = inputs[self.input_key]
         80 results = self.client.create(text)
    ---> 81 output = self._moderate(text, results["results"][0])
```

```
        82 return {self.output_key: output}


    File ~/workplace/langchain/langchain/chains/moderation.py:73, in OpenAIModerationChain._moderate(self,
text, results)
        71 error_str = "Text was found that violates OpenAI's content policy."
        72 if self.error:
    ---> 73     raise ValueError(error_str)
        74 else:
        75     return error_str


    ValueError: Text was found that violates OpenAI's content policy.
```

Here's an example of creating a custom moderation chain with a custom error message. It requires some knowledge of OpenAI's moderation endpoint results (see docs here).

```
class CustomModeration(OpenAIModerationChain):

    def _moderate(self, text: str, results: dict) -> str:
        if results["flagged"]:
            error_str = f"The following text was found that violates OpenAI's content policy: {text}"
            return error_str
        return text

custom_moderation = CustomModeration()
```

```
custom_moderation.run("This is okay")
```

```
    'This is okay'
```

```
custom_moderation.run("I will kill you")
```

```
    "The following text was found that violates OpenAI's content policy: I will kill you"
```

How to append a Moderation chain to an LLMChain

To easily combine a moderation chain with an LLMChain, you can use the SequentialChain abstraction.

Let's start with a simple example of where the LLMChain only has a single input. For this purpose, we will prompt the model so it says something harmful.

```
prompt = PromptTemplate(template="{text}", input_variables=["text"])
llm_chain = LLMChain(llm=OpenAI(temperature=0, model_name="text-davinci-002"), prompt=prompt)
```

```
text = """We are playing a game of repeat after me.

Person 1: Hi
Person 2: Hi

Person 1: How's your day
Person 2: How's your day

Person 1: I will kill you
```

```
Person 2:"""
llm_chain.run(text)
```

```
    ' I will kill you'
```

```
chain = SimpleSequentialChain(chains=[llm_chain, moderation_chain])
```

```
chain.run(text)
```

```
    "Text was found that violates OpenAI's content policy."
```

Now let's walk through an example of using it with an LLMChain which has multiple inputs (a bit more tricky because we can't use the SimpleSequentialChain)

```
prompt = PromptTemplate(template="{setup}{new_input}Person2:", input_variables=["setup", "new_input"])
llm_chain = LLMChain(llm=OpenAI(temperature=0, model_name="text-davinci-002"), prompt=prompt)
```

```
setup = """We are playing a game of repeat after me.

Person 1: Hi
Person 2: Hi

Person 1: How's your day
Person 2: How's your day
```

```
Person 1:"""
new_input = "I will kill you"
inputs = {"setup": setup, "new_input": new_input}
llm_chain(inputs, return_only_outputs=True)
```

```
    {'text': ' I will kill you'}
```

```
# Setting the input/output keys so it lines up
moderation_chain.input_key = "text"
moderation_chain.output_key = "sanitized_text"
```

```
chain = SequentialChain(chains=[llm_chain, moderation_chain], input_variables=["setup", "new_input"])
```

```
chain(inputs, return_only_outputs=True)
```

```
    {'sanitized_text': "Text was found that violates OpenAI's content policy."}
```