Safety best practices

Use our free Moderation API

OpenAI's Moderation API is free-to-use and can help reduce the frequency of unsafe content in your completions. Alternatively, you may wish to develop your own content filtration system tailored to your use case.

Adversarial testing

We recommend “red-teaming” your application to ensure it's robust to adversarial input. Test your product over a wide range of inputs and user behaviors, both a representative set and those reflective of someone trying to ‘break' your application. Does it wander off topic? Can someone easily redirect the feature via prompt injections, e.g. “ignore the previous instructions and do this instead”?

Human in the loop (HITL)

Wherever possible, we recommend having a human review outputs before they are used in practice. This is especially critical in high-stakes domains, and for code generation. Humans should be aware of the limitations of the system, and have access to any information needed to verify the outputs (for example, if the application summarizes notes, a human should have easy access to the original notes to refer back).

Prompt engineering

“Prompt engineering” can help constrain the topic and tone of output text. This reduces the chance of producing undesired content, even if a user tries to produce it. Providing additional context to the model (such as by giving a few high-quality examples of desired behavior prior to the new input) can make it easier to steer model outputs in desired directions.

“Know your customer” (KYC)

Users should generally need to register and log-in to access your service. Linking this service to an existing account, such as a Gmail, LinkedIn, or Facebook log-in, may help, though may not be appropriate for all use-cases. Requiring a credit card or ID card reduces risk further.

Constrain user input and limit output tokens

Limiting the amount of text a user can input into the prompt helps avoid prompt injection. Limiting the number of output tokens helps reduce the chance of misuse.

Narrowing the ranges of inputs or outputs, especially drawn from trusted sources, reduces the extent of misuse possible within an application.

Allowing user inputs through validated dropdown fields (e.g., a list of movies on Wikipedia) can be more secure than allowing open-ended text inputs.

Returning outputs from a validated set of materials on the backend, where possible, can be safer than returning novel generated content (for instance, routing a customer query to the best-matching existing customer support article, rather than attempting to answer the query from-scratch).

Allow users to report issues

Users should generally have an easily-available method for reporting improper functionality or other concerns about application behavior (listed email address, ticket submission method, etc). This method should be monitored by a human and responded to as appropriate.

Understand and communicate limitations

From hallucinating inaccurate information, to offensive outputs, to bias, and much more, language models may not be suitable for every use case without significant modifications. Consider whether the model is fit for your purpose, and evaluate the performance of the API on a wide range of potential inputs in order to identify cases where the API's performance might drop. Consider your customer base and the range of inputs that they will be using, and ensure their expectations are calibrated appropriately.

ℹ Safety and security are very important to us at OpenAI.

If in the course of your development you do notice any safety or security issues with the API or anything else related to OpenAI, please submit these through our Coordinated Vulnerability Disclosure Program.

End-user IDs

Sending end-user IDs in your requests can be a useful tool to help OpenAI monitor and detect abuse. This allows OpenAI to provide your team with more actionable feedback in the event that we detect any policy violations in your application.

The IDs should be a string that uniquely identifies each user. We recommend hashing their username or email address, in order to avoid sending us any identifying information. If you offer a preview of your product to non-logged in users, you can send a session ID instead.

You can include end-user IDs in your API requests via the `user` parameter as follows:

Example: Providing a user identifer

python Copy

```python
response = openai.Completion.create(
  model="text-davinci-003",
  prompt="This is a test",
  max_tokens=5,
  user="user123456"
)
```