QuickStart: Analyze text content

Article • 10/16/2023

Get started with the Content Safety Studio, REST API, or client SDKs to do basic text moderation. The Azure AI Content Safety service provides you with AI algorithms for flagging objectionable content. Follow these steps to try it out.

> **Note**
>
> The sample data and code may contain offensive content. User discretion is advised.

Reference documentation | Library source code | Package (npm) | Samples |

Prerequisites

- An Azure subscription - Create one for free
- The current version of Node.js
- Once you have your Azure subscription, create a Content Safety resource in the Azure portal to get your key and endpoint. Enter a unique name for your resource, select the subscription you entered on the application form, select a resource group, supported region, and supported pricing tier. Then select **Create**.
  - The resource takes a few minutes to deploy. After it finishes, Select **go to resource**. In the left pane, under **Resource Management**, select **Subscription Key and Endpoint**. The endpoint and either of the keys are used to call APIs.

Set up application

Create a new Node.js application. In a console window (such as cmd, PowerShell, or Bash), create a new directory for your app, and navigate to it.

Console

```
mkdir myapp && cd myapp
```

Run the `npm init` command to create a node application with a `package.json` file.

Console

```
npm init
```

Install the client SDK

Install the `@azure-rest/ai-content-safety` npm package:

Console

```
npm install @azure-rest/ai-content-safety
```

Also install the `dotenv` module to use environment variables:

Console

```
npm install dotenv
```

Your app's `package.json` file will be updated with the dependencies.

Create environment variables

In this example, you'll write your credentials to environment variables on the local machine running the application.

> **Tip**
>
> Don't include the key directly in your code, and never post it publicly. See the Azure AI services **security** article for more authentication options like **Azure Key Vault**.

To set the environment variable for your key and endpoint, open a console window and follow the instructions for your operating system and development environment.

1. To set the `CONTENT_SAFETY_KEY` environment variable, replace `YOUR_CONTENT_SAFETY_KEY` with one of the keys for your resource.
2. To set the `CONTENT_SAFETY_ENDPOINT` environment variable, replace `YOUR_CONTENT_SAFETY_ENDPOINT` with the endpoint for your resource.

Windows

Console

```
setx CONTENT_SAFETY_KEY 'YOUR_CONTENT_SAFETY_KEY'
```

Console

```
setx CONTENT_SAFETY_ENDPOINT 'YOUR_CONTENT_SAFETY_ENDPOINT'
```

After you add the environment variables, you might need to restart any running programs that will read the environment variables, including the console window.

Analyze text content

Create a new file in your directory, *index.js*. Open it in your preferred editor or IDE and paste in the following code. Replace `<your text sample>` with the text content you'd like to use.

Tip

Text size and granularity

The default maximum length for text submissions is **10K** characters.

JavaScript

```
const ContentSafetyClient = require("@azure-rest/ai-content-safety").default,
  { isUnexpected } = require("@azure-rest/ai-content-safety");
const { AzureKeyCredential } = require("@azure/core-auth");

// Load the .env file if it exists
require("dotenv").config();

async function main() {
    // get endpoint and key from environment variables
    const endpoint = process.env["CONTENT_SAFETY_ENDPOINT"];
    const key = process.env["CONTENT_SAFETY_KEY"];

    const credential = new AzureKeyCredential(key);
```

```
    const client = ContentSafetyClient(endpoint, credential);

    // replace with your own sample text string
    const text = "<your sample text>";
    const analyzeTextOption = { text: text };
    const analyzeTextParameters = { body: analyzeTextOption };

    const result = await
client.path("/text:analyze").post(analyzeTextParameters);

    if (isUnexpected(result)) {
        throw result;
    }

    console.log("Hate severity: ", result.body.hateResult?.severity);
    console.log("SelfHarm severity: ", result.body.selfHarmResult?.severity);
    console.log("Sexual severity: ", result.body.sexualResult?.severity);
    console.log("Violence severity: ", result.body.violenceResult?.severity);
}

main().catch((err) => {
    console.error("The sample encountered an error:", err);
});
```

Run the application with the `node` command on your quickstart file.

Console

```
node index.js
```

Output

Console

```
Hate severity:  0
SelfHarm severity:  0
Sexual severity:  0
Violence severity:  0
```

Clean up resources

If you want to clean up and remove an Azure AI services subscription, you can delete the resource or resource group. Deleting the resource group also deletes any other resources

associated with it.

- Portal
- Azure CLI

Next steps

Configure filters for each category and test on datasets using Content Safety Studio, export the code and deploy.

Content Safety Studio quickstart