

UNIVERSITY OF OSLO

The ChatGPT API: What, why, how@Uio

Knut Waagan, Senior Engineer, IT Department
Sherin Sugathan, Senior Engineer, dScience

Oct 8, 2024



What is an API?

- gpt.uio.no is a ***user interface***
- It is designed for human interaction with a computer
- What if we want a computer code that prompts GPT and then manages (e.g. stores) the response?
- Then an ***application programming interface*** is more practical
- How does it work? We send and receive data in a strictly defined format

✧ GPT-4

☰ Velg instruksjon



Er det noe jeg kan hjelpe deg med?

For å starte en samtale, kan du skrive inn spørsmålet ditt eller be om hjelp med et bestemt tema. Du kan også finne inspirasjon i noen av eksemplene nedenfor som kan hjelpe deg i gang.

💡 Forslag til bruk

“Kan du gi meg en kortfattet oppsummering av en tekst? Jeg sender teksten i neste melding.”

“Kan du hente de viktigste stikkordene fra en tekst og returnere dem som en kommaseparert liste? Jeg sender teksten i neste melding.”

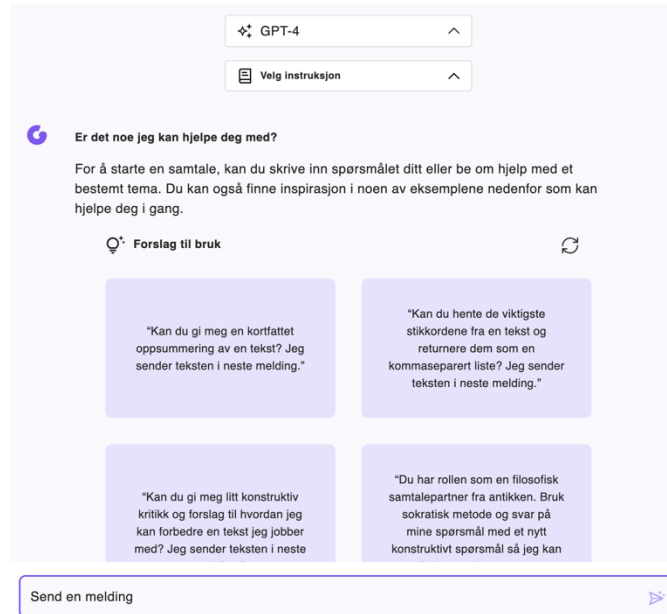
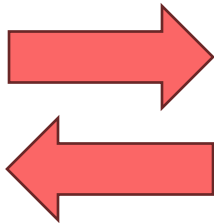
“Kan du gi meg litt konstruktiv kritikk og forslag til hvordan jeg kan forbedre en tekst jeg jobber med? Jeg sender teksten i neste melding.”

“Du har rollen som en filosofisk samtalepartner fra antikken. Bruk den sokratiske metode og svar på mine spørsmål med et nytt og konstruktivt spørsmål så jeg kan fortsette å tenke videre.”

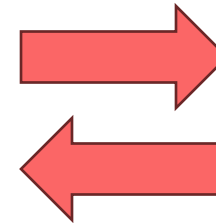
Send en melding

How gpt.uio.no works

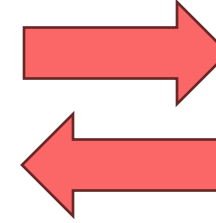
User



The screenshot shows the gpt.uio.no interface. At the top, there's a dropdown menu set to 'GPT-4' and another labeled 'Velg instruksjon'. Below this, a purple 'G' icon is followed by the text 'Er det noe jeg kan hjelpe deg med?'. A paragraph explains that users can start a conversation by writing a question or asking for help on a specific topic, and that there are examples below for inspiration. A section titled 'Forslag til bruk' (Suggestions for use) contains four purple boxes with prompts: 'Kan du gi meg en kortfattet oppsummering av en tekst? Jeg sender teksten i neste melding.', 'Kan du hente de viktigste stikkordene fra en tekst og returnere dem som en kommaseparert liste? Jeg sender teksten i neste melding.', 'Kan du gi meg litt konstruktiv kritikk og forslag til hvordan jeg kan forbedre en tekst jeg jobber med? Jeg sender teksten i neste melding.', and 'Du har rollen som en filosofisk samtalepartner fra antikken. Bruk sokratisk metode og svar på mine spørsmål med et nytt konstruktivt spørsmål så jeg kan'. At the bottom is a text input field with the placeholder 'Send en melding' and a send button.



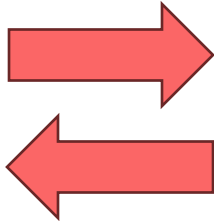
API



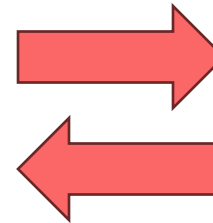
ChatGPT

How our code might work

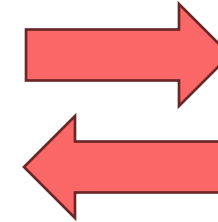
Prog-
rammer



```
response = openai.ChatCompletion.create(  
    engine="cjeu-4",  
    messages=[{"role": "system", "content": system_prompt},  
              {"role": "user", "content": theprompt}],  
    temperature=0.0,  
    max_tokens=3000,  
    top_p=1,  
    frequency_penalty=0,  
    presence_penalty=0,  
    stop=None)
```



API



ChatGPT

Why bother? A use case

Scanned image

ation

o . . .

I — *Translator's note:* This is a corrected version of the text appearing in the Official Journal, which is defective: cf. Mr Advocate General Warner's comments in Case 29/77 *Roquette* 1977 ECR 1835, at p.1847.

After OCR

"I - Tram/,11or's notr: This is :t 1.:orrtetC'd ,rsion of the teiu
.,prC',uing in tlu- Offidal Journ.11, which is ddccm•e;
1.f. !\fr :~dn,;,;,;.nC' Grnrrr;l ~:irorr'i .:omml."nu u1 Case
29177 Roqum, 1977 ECR IHJS, Jt p.18•7."

Can ChatGPT correct the OCR? Thousands of scans?

Why bother? More reasons

- Longer prompts
- Tweak parameters, e.g. Randomness
- Additional functionality, e.g. fine-tuning certain models

Deployment * [+ Create new deployment](#) ▾

gpt-4o-mini (version:2024-07-18) ▾

System message Add your data **Parameters**

Past messages included ⓘ

10

Max response ⓘ

800

Temperature ⓘ

0.7

Top P ⓘ

0.95

Stop sequence ⓘ

Frequency penalty ⓘ

0

Presence penalty ⓘ

0

Why use the API offered at UiO?

- In short: For privacy, and for legal reasons
- OpenAI has a public API, but it is not approved for use at UiO
- UiO has agreement with Microsoft that ensures GDPR etc
- Azure's API to GPT is included
- Prompts and responses are not stored, and not used for training AI models
- Not for sensitive data, only   i.e. same restrictions as gpt.uio.no
- For sensitive data, perhaps LLM running locally soon?
- One further restriction: Microsoft's «content filtering» to prevent «potential harmful content». It might affect certain research topics

The use case: OCR scans

- Researchers: «We want to correct OCR scans with GPT»
- OCR = **Optical Character Recognition**
- OCR makes errors, can we prompt GPT to restore the original text?
- Raw material: EU law documents

Dr Michal Ovádek



Lecturer in European Institutions, Politics and Policy

Room: 2.02 36-38 Gordon Square

Email: m.ovadek@ucl.ac.uk

[Twitter](#)

[Website](#)

COMMISSION v LUXEMBOURG AND BELGIUM

1958, of a special duty leviable upon the issue of import licences for certain milk products,

THE COURT

composed of: Ch. L. Hammes, President, A. M. Donner (Rapporteur) and R. Lecourt, Presidents of Chambers, L. Delvaux and A. Trabucchi, Judges,

Advocate-General: K. Roemer

Registrar: A. Van Houtte

gives the following

JUDGMENT

Issues of fact and of law

I—Facts and procedure

It is common ground that by Royal Decree of 3 November 1958 and by Grand-Ducal Decree of 17 November

these observations, issued a reasoned opinion dated 3 April 1963 under the first paragraph of Article 169 of the Treaty stating that the two governments had failed to fulfil their obligations



Daniel Naurin

Director and Professor of Political Science - ARL
Centre for European Studies

Which OCR?

- OCR engines differ
- Some recognise characters well
- ... but struggle with reading order and column separation
- Loooong story short: Paragraph order crucial, hence prioritised

1958, of a special duty leviable upon the issue of import licences for certain milk products,

THE COURT

composed of: Ch. L. Hammes, President, A. M. Donner (Rapporteur) and R. Lecourt, Presidents of Chambers, L. Delvaux and A. Trabucchi, Judges,

Advocate-General: K. Roemer
Registrar: A. Van Houtte

gives the following

JUDGMENT

Issues of fact and of law

I—Facts and procedure

It is common ground that by Royal Decree of 3 November 1958 and by Grand-Ducal Decree of 17 November 1958 the Belgian and Luxembourg governments introduced a duty levied on the issue of import licences for certain milk products. Maximum rates were fixed by these decrees, and the amounts actually charged were fixed in each case by a ministerial order made on the same day. These duties were later amended by both governments by a long series of orders fixing the maximum rates and the rates actually charged.

The Commission took the view that these measures, which applied to the importing of products originating in Member States or in free circulation in those States, were contrary to Article 12 of the Treaty. So by letter of 8 November 1961 it expressed its disapproval of the maintenance of these measures to the Belgian and Luxembourg governments, and invited them to present their observations on this matter. On 19 April 1963 the Commission, after receiving

these observations, issued a reasoned opinion dated 3 April 1963 under the first paragraph of Article 169 of the Treaty stating that the two governments had failed to fulfil their obligations arising out of Article 12 of the Treaty and inviting them to take the measures necessary to comply with it within a period of one month.

By letter of 8 May 1963 the Belgian government stated that it agreed 'to abolish the duties in force on licences for milk products so soon as an adequate substitute formula is established by agreement with the Commission'. By letter of 9 May 1963 the Luxembourg government stated that it supported the attitude adopted by the Belgian government for the products in question.

Since the levying of the duties in question was continued the Commission instituted the present proceedings on 15 October 1963.

By order dated 28 November 1963 the Court joined the two cases as having related subject-matter.

The procedure followed the normal course.

Which OCR?

- OCR engines differ
- Some recognise characters well
- ... but struggle with reading order and column separation
- Loooong story short: Paragraph order crucial, hence prioritised
- This left some work for GPT:

Translator's note: This is a corrected version of the text appearing in the Official Journal, which is derived from Mr Advocate General Warner's comments in 29/77 Roquette 1977 ECR I-1835, at p.1847.

SH

You

Correct the following text

"I - Tram/,1.1or's notr: This is :t 1.-orrtetC'd ,;rsion of the teiu
.,prC',uing in tlu- Offidal Journ.11, which is ddccm•e;
1.-f. !\fr :~dn,;,,,;.nC' Grnrrr;l ~:irorr'i .:omml."nu u1 Case
29177 Roqum, 1977 ECR IHJS, Jt p.18•7."

Use case: Test result GPT-4



You

Correct the following text

"I - Translator's note: This is a corrected version of the text appearing in the Official Journal, which is defective; ref. Mr. Advocate General Warner's comments on Case 29177 Roquette, 1977 ECR 1885, at p. 1887."



ChatGPT

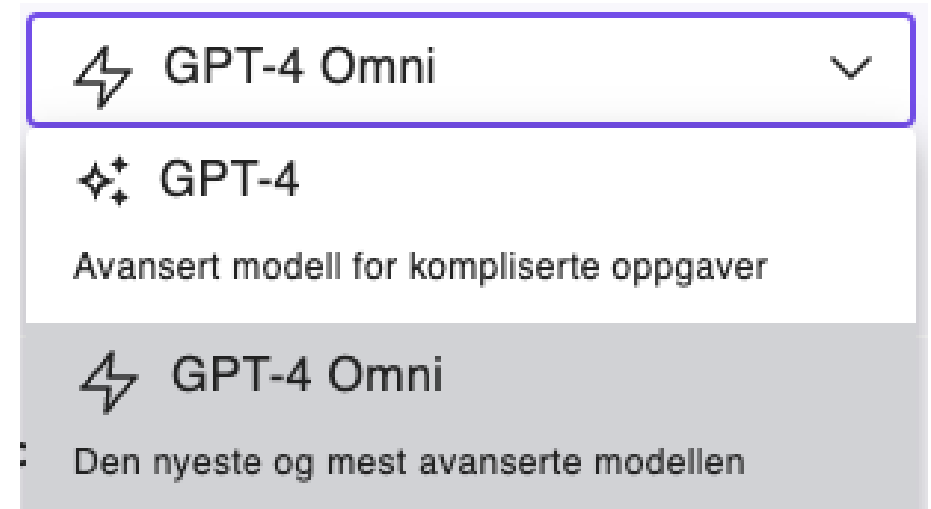
Certainly! Here's the corrected version of the provided text:

"I - Translator's note: This is a corrected version of the text appearing in the Official Journal, which is defective; ref. Mr. Advocate General Warner's comments on Case 29177 Roquette, 1977 ECR 1885, at p. 1887."

I — *Translator's note: This appearing in the Official Journal, which is defective; ref. Mr. Advocate General Warner's comments on Case 29177 Roquette 1977*

Model choice

- Multiple models: GPT-4, GPT-4 Omni (aka GPT-4o), GPT-3.5 Turbo
- Differ in **quality, speed, cost, context window**
- Context window = How much text that the model can deal with at once
- When in doubt, test



Model choice on gpt.uio.no

Price is per token

- LLMs operate on (and generate) a sequence of **tokens**
- «optical character recognition» is 4 tokens to GPT-4o
- «optical» gets split into «opt» and «ical»
- Thumb rule English: 100 tokens per 75 words
- Tip: Count tokens with <https://github.com/openai/tiktoken>
- Price list: <https://azure.microsoft.com/en-us/pricing/details/cognitive-services/openai-service/>

Some prices early Oct

Modeller	Kontekst	Input (Per 1,000,000 tokens)	Output (Per 1,000,000 tokens)
GPT-3.5-Turbo-0301	4K	\$2	\$2
GPT-3.5-Turbo-0613	4K	\$1,50	\$2
GPT-3.5-Turbo-0613	16K	\$3	\$4
GPT-3.5-Turbo-1106	16K	\$1	\$2
GPT-3.5-Turbo-0125	16K	\$0,50	\$1,50
GPT-3.5-Turbo-Instruct	4K	\$1,50	\$2
GPT-4-Turbo	128K	\$10	\$30
GPT-4-Turbo-Vision	128K	\$10	\$30
GPT-4	8K	\$30	\$60
GPT-4	32K	\$60	\$120

Some prices early Oct

Modeller	Kontekst	Input (Per 1,000,000 tokens)	Output (Per 1,000,000 tokens)
o1 preview Global Deployment	128K	\$15	\$60
o1 preview Regional API	128K	\$16,50	\$66
o1 mini Global Deployment	128K	\$3	\$12
o1 mini Regional API	128K	\$3,30	\$13,20
gpt-4o-2024-08-06 Global Deployment	128K	\$2,50	\$10
gpt-4o-2024-08-06 Regional API	128K	\$2,75	\$11
Global utrulling av GPT-4o	128K	\$5	\$15
GPT-4o Regional API	128K	\$5	\$15
GPT-4o-mini global utrulling	128K	\$0,15	\$0,60
GPT-4o-mini regional API	128K	\$0,165	\$0,66

How to apply for API access

- Fill in form at <https://www-int.uio.no/tjenester/it/ki/gpt-uio/bestill-api/>
- Provide short project description and billing info etc
- Wait for email with link to your «subscription»
- Click your way to «Azure OpenAI»...

[Home](#) > [Azure AI services](#)



Azure AI services | Azure OpenAI ✨ ...

Azure AI services

Search



Manage deleted resources



Manage view



Refresh



Export to CSV



Open query



Assign tags



Delete

Overview

All Azure AI services

Azure AI services

Azure AI services

Azure OpenAI

AI Search

Computer vision

Face API

Custom vision

Speech service

Language service

Filter for any field...

Subscription equals all

Type equals all

Resource group equals all



Location equals all



Add filter

Showing 1 to 1 of 1 records.

No grouping



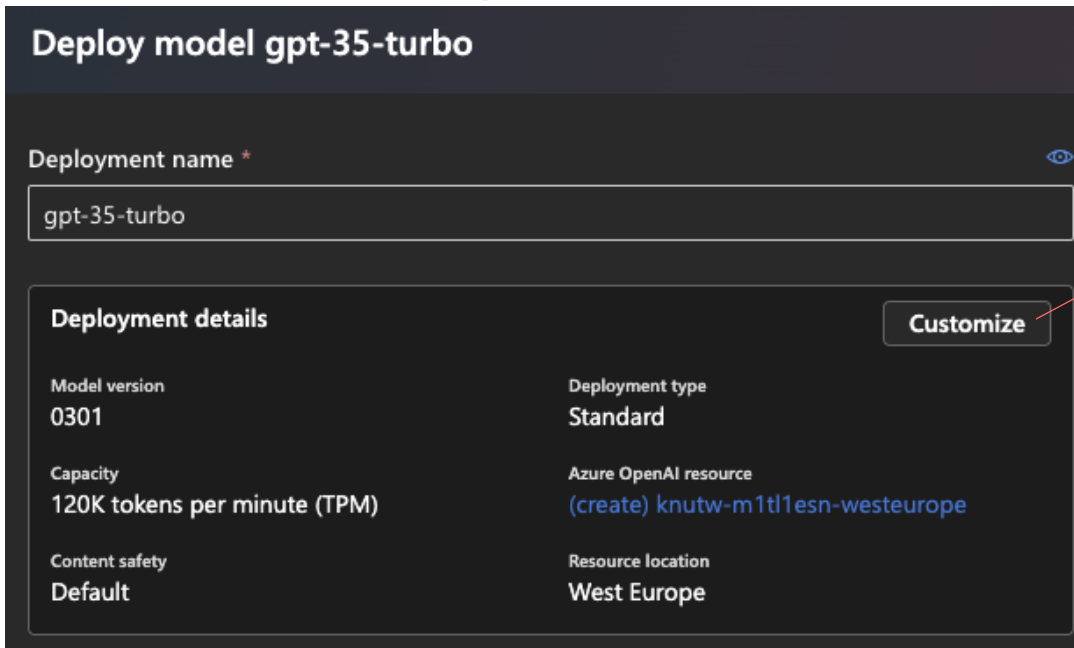
List view



☐ Name ↑↓	Kind ↑↓	Location ↑↓	Custom Domain ... ↑↓	Pricing tier ↑↓	Status ↑↓	Created date ↑↓
☐ gpt-usit-ffu-sv-ki-research-swe-1	OpenAI	Sweden Central	gpt-usit-ffu-sv-ki-...	S0	Succeeded	2024-08-20T13:38:02.... ⋮

How to set up a model on Azure

- Click on the «Resource group» (In our case named `gpt_usit_ffu_sv_ki_research`)
- Click «Model deployments» -> Manage Deployments
- Click on Chat -> [+ Create new deployment](#)
- Select model
- Deployment type: Standard (Certainly not Global)
- Location: In Europe



Deploy model gpt-35-turbo

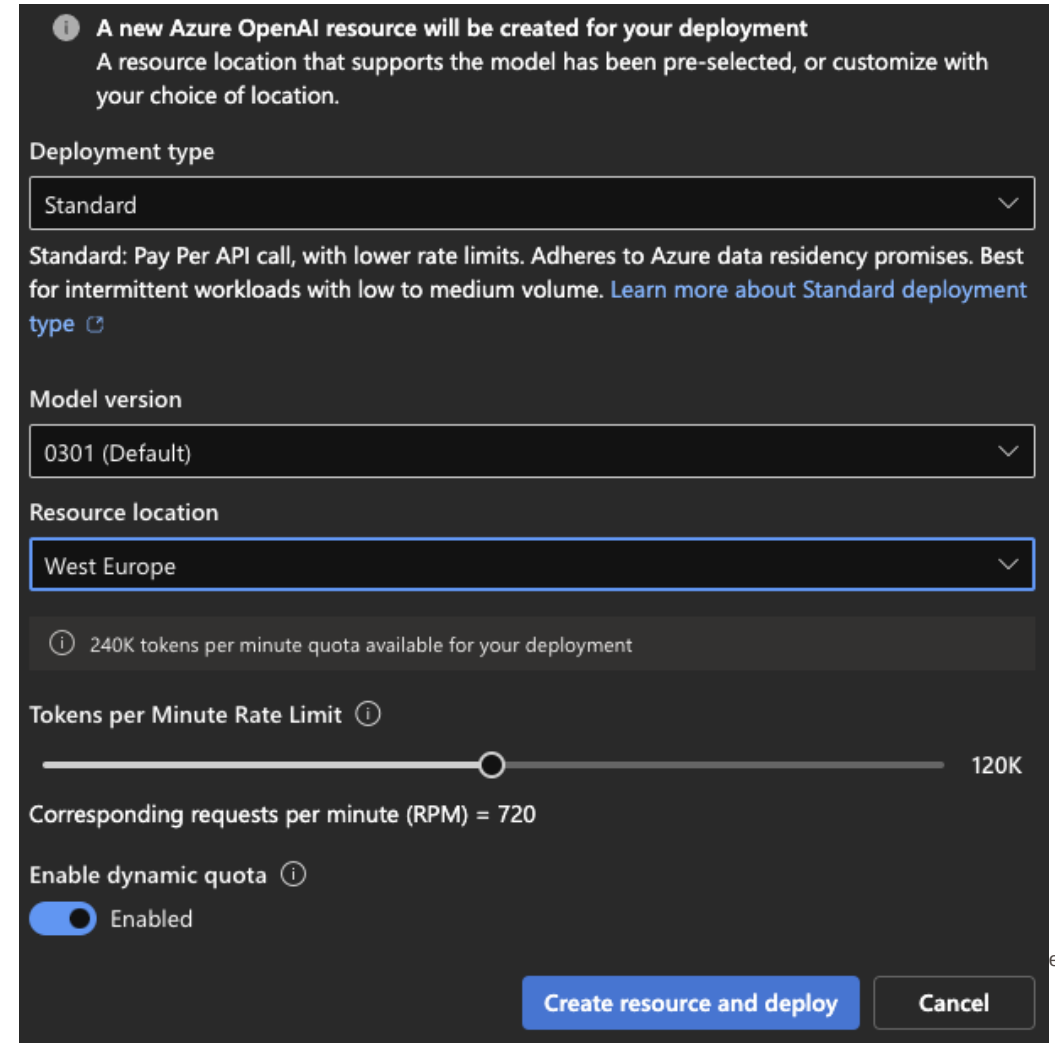
Deployment name *

gpt-35-turbo

Deployment details

Model version	0301	Deployment type	Standard
Capacity	120K tokens per minute (TPM)	Azure OpenAI resource	(create) knutw-m1t1esn-westeurope
Content safety	Default	Resource location	West Europe

[Customize](#)



A new Azure OpenAI resource will be created for your deployment
A resource location that supports the model has been pre-selected, or customize with your choice of location.

Deployment type

Standard

Standard: Pay Per API call, with lower rate limits. Adheres to Azure data residency promises. Best for intermittent workloads with low to medium volume. [Learn more about Standard deployment type](#)

Model version

0301 (Default)

Resource location

West Europe

240K tokens per minute quota available for your deployment

Tokens per Minute Rate Limit

120K

Corresponding requests per minute (RPM) = 720

Enable dynamic quota

Enabled

[Create resource and deploy](#) [Cancel](#)

Generate basic Python code and test it

- Use the «Chat playground»
- Optional system message

Chat playground

</> View code 🔧 Deploy ▾ →| Import ▾ |→ Export ▾

Setup ⊞ Hide

Deployment * + Create new deployment ▾

gpt-4o-mini (version:2024-07-18) ▾

System message

Add your data Parameters

Give the model instructions and context ⓘ ↶

You are an AI assistant that helps people find information.

Generate basic Python code

- Use the «Chat» tool
- Optional system message
- Parameter:
<https://learn.microsoft.com/en-us/azure/ai-services/openai/reference#request-body>

System message

Add your data

Parameters

Past messages
included ⓘ

 10

Max response ⓘ

 4000

Temperature ⓘ

 0.7

Top P ⓘ

 0.95

Stop sequence ⓘ

Frequency penalty ⓘ

 0

Presence penalty ⓘ

 0

Generate basic Python code and test it

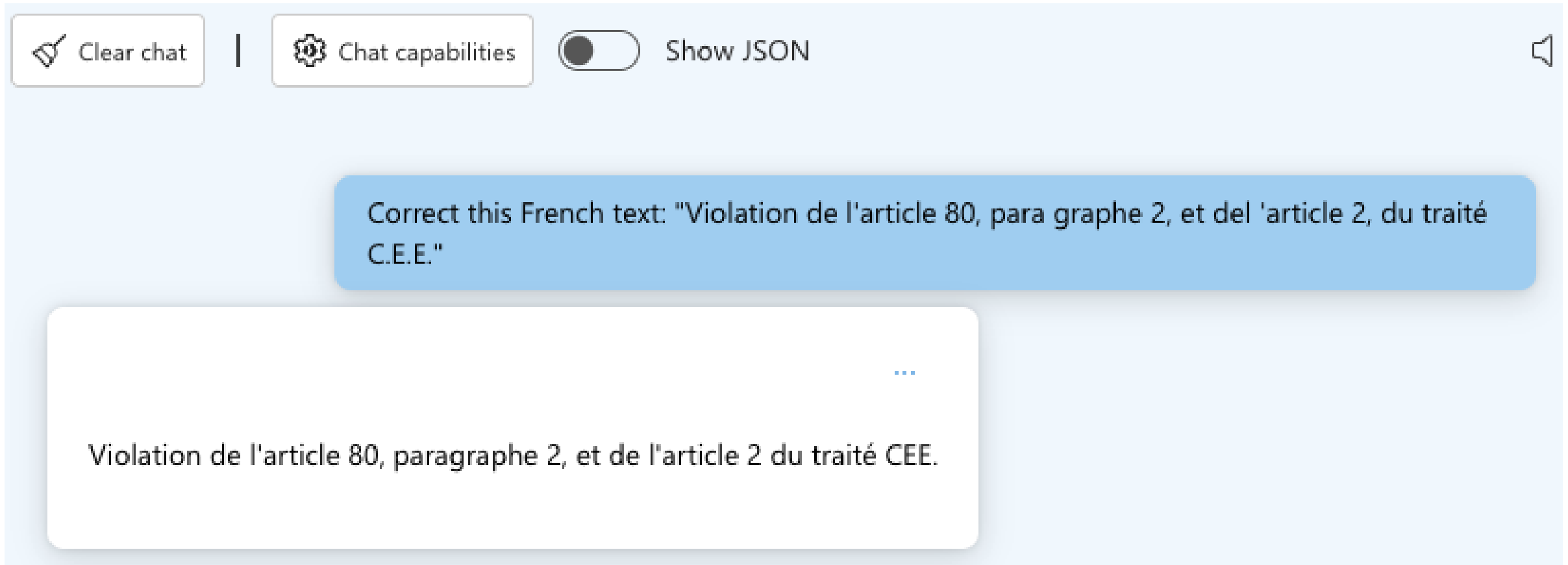
- Type your prompt

Correct this French text: "Violation de l'article 80, para graphe 2, et del 'article 2, du traité C.E.E."

35/4000 tokens to be sent  

Generate basic Python code and test it

- Check response



The screenshot shows a chat interface with a light blue background. At the top, there is a header bar containing three elements: a 'Clear chat' button with a trash icon, a 'Chat capabilities' button with a gear icon, and a 'Show JSON' toggle switch which is currently turned off. A speaker icon is located on the far right of the header. The chat area contains two messages. The first message is a blue bubble with the text: 'Correct this French text: "Violation de l'article 80, para graphe 2, et del 'article 2, du traité C.E.E."''. The second message is a white bubble with the text: 'Violation de l'article 80, paragraphe 2, et de l'article 2 du traité CEE.' and a blue ellipsis icon at the end.

Clear chat | Chat capabilities Show JSON

Correct this French text: "Violation de l'article 80, para graphe 2, et del 'article 2, du traité C.E.E."

...

Violation de l'article 80, paragraphe 2, et de l'article 2 du traité CEE.

Generate basic Python code and test it

- JSON format



Clear chat



Chat capabilities



Show JSON

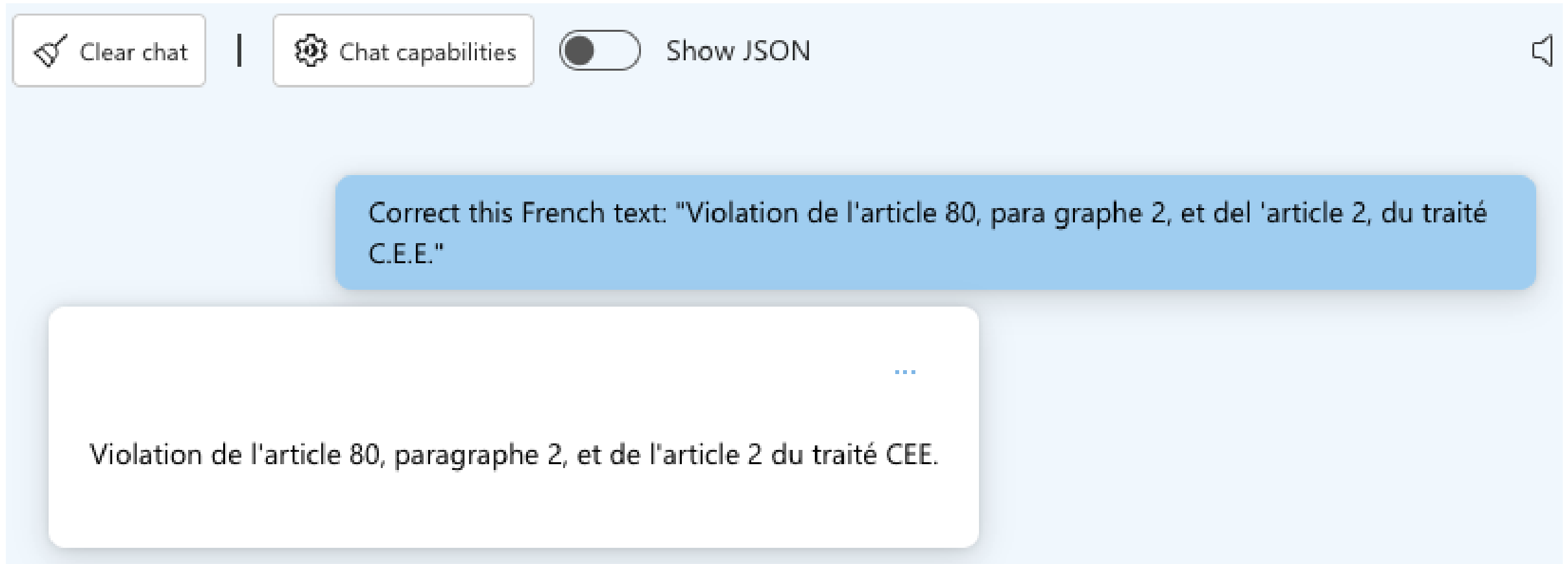


This view shows how the chat transcript is formatted for the API.

```
1  [
2    {
3      "role": "user",
4      "content": "Correct this French text: \"Violation de l'article 80, para graphe 2, et
5      },
6      {
7      "role": "assistant",
8      "content": "Violation de l'article 80, paragraphe 2, et de l'article 2 du traité CEE."
9    }
10 ]
```

Generate basic Python code and test it

- Check response



Generate basic

- Click `</>` View code
- Python is default option, also JSON, C# and Curl
- Python package openai installed via e.g. Pip
- There is a similar package for R

Sample Code

You can use the following code to start integrating your current prompt and settings into your application.

<https://knutw-m1uon487-westeuropa.openai.azure.com/>

python ▾

Entra ID authentication

Key authentication

```
1 import os
2 from openai import AzureOpenAI
3
4 endpoint = os.getenv("ENDPOINT_URL", "https://knutw-m1uon487-westeuropa.openai.azure.com")
5 deployment = os.getenv("DEPLOYMENT_NAME", "gpt-35-turbo")
6 subscription_key = os.getenv("AZURE_OPENAI_API_KEY", "REPLACE_WITH_YOUR_KEY_VALUE_HERE")
7
8 # Initialize Azure OpenAI client with key-based authentication
9 client = AzureOpenAI(
10     azure_endpoint = endpoint,
11     api_key = subscription_key,
12     api_version = "2024-05-01-preview",
13 )
14
15 completion = client.chat.completions.create(
16     model=deployment,
17     messages= [
18         {
19             "role": "user",
20             "content": "Correct this French text: \"Violation de l'article 80, para graphe 2"
```

Generate bas

- Prompt
- Response also included
- Parameters

```
1 import os
2 from openai import AzureOpenAI
3
4 endpoint = os.getenv("ENDPOINT_URL", "https://knutw-m1uon487-westeuropa.openai.azure.com")
5 deployment = os.getenv("DEPLOYMENT_NAME", "gpt-35-turbo")
6 subscription_key = os.getenv("AZURE_OPENAI_API_KEY", "REPLACE_WITH_YOUR_KEY_VALUE_HERE")
7
8 # Initialize Azure OpenAI client with key-based authentication
9 client = AzureOpenAI(
10     azure_endpoint = endpoint,
11     api_key = subscription_key,
12     api_version = "2024-05-01-preview",
13 )
14
15 completion = client.chat.completions.create(
16     model=deployment,
17     messages= [
18         {
19             "role": "user",
20             "content": "Correct this French text: \"Violation de l'article 80, para graphe 2"
21         },
22         {
23             "role": "assistant",
24             "content": "Violation de l'article 80, paragraphe 2, et de l'article 2 du traité"
25         }
26     ],
27     past_messages=10,
28     max_tokens=4000,
29     temperature=0.7,
30     top_p=0.95,
31     frequency_penalty=0,
```

Generate basic

- **API key** hidden from view
- This is like a password for using your API instance, so be careful!

```
20     "content": "Correct this French text: \"Violation de l'article 80, para graphe 2  
21     },  
22     {  
23         "role": "assistant",  
24         "content": "Violation de l'article 80, paragraphe 2, et de l'article 2 du traité  
25     }  
26 ],  
27     past_messages=10,  
28     max_tokens=4000,  
29     temperature=0.7,  
30     top_p=0.95,  
31     frequency_penalty=0,  
32     presence_penalty=0,  
33     stop=None,  
34     stream=False  
35 )  
36  
37 print(completion.to_json())
```

You should use environment variables or a secret management tool like Azure Key Vault to prevent accidental exposure of your key in applications.

[Learn more about setting up an environment](#)

Endpoint ⓘ

API key ⓘ

Generate basic Python

- Here is what our test code prints
- The API returns all this
- GPT's response
- E.g. info about filtering
- Possible to ask for multiple responses, hence they are listed in the «choices» object

```
{
  "id": "chatcmpl-AEcaaWAhHQV5JL93kHDXCMNA6EvAU",
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "Violation de l'article 80, paragraphe 2, et de",
        "role": "assistant"
      },
      "content_filter_results": {
        "hate": {
          "filtered": false,
          "severity": "safe"
        },
        "self_harm": {
          "filtered": false,
          "severity": "safe"
        },
        "sexual": {
          "filtered": false,
          "severity": "safe"
        },
        "violence": {
          "filtered": false,
```

Generate basic Python

- Also how many tokens in prompt and response

```
    "violence": {
      "filtered": false,
      "severity": "safe"
    }
  },
  "created": 1728048292,
  "model": "gpt-35-turbo",
  "object": "chat.completion",
  "system_fingerprint": null,
  "usage": {
    "completion_tokens": 27,
    "prompt_tokens": 43,
    "total_tokens": 70
  },
  "prompt_filter_results": [
    {
      "prompt_index": 0,
      "content_filter_results": {}
    }
  ]
}
```

The openai package

- The openai Python package takes care of the details
- We do not have to learn all the API specifics
- But probably possible interact more directly with the API

What about multiple scans?

- For each scan we need to
 1. Read the text from file
 2. Compose a prompt with instruction + data
 3. Call the API
 4. Store the result
- Program a loop. Or ask an LLM to do it?

Scaling Up: Large-scale text processing with Azure API

Problem

Perform corrections on ~55000 text files using Azure API

55000 pages x ~110 seconds = 6,050,000 seconds = ~ 70 days

Challenges

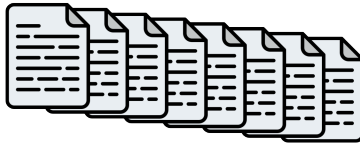
Serial execution will take more time

Errors from API when we issue too many requests

Need to deliver fast

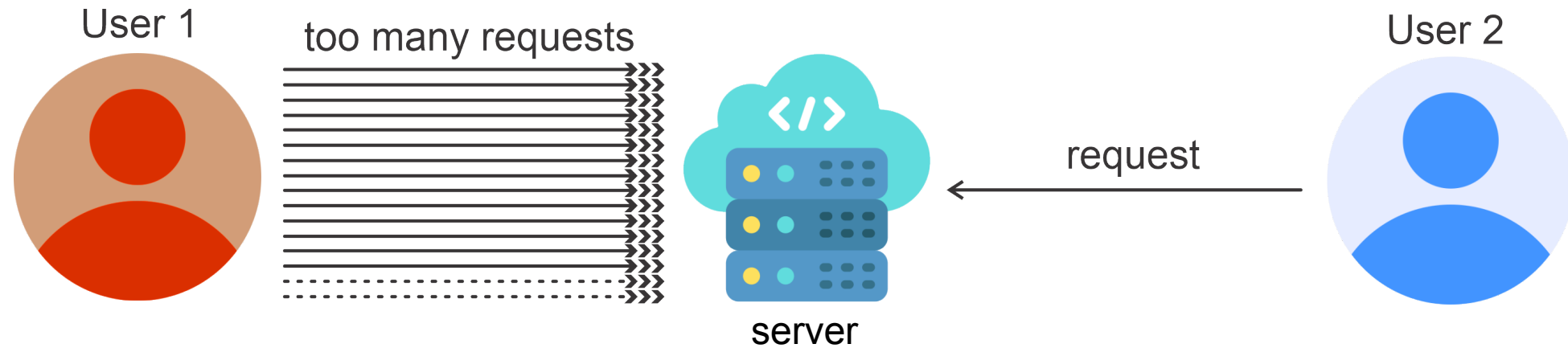
Code bugs can result in high costs

Scaling Up: Large-scale text processing with Azure API

```
loop ( {  
    1. Read next file from disk  
    2. Call API  
}
```

Executes serially and is prone to hitting rate limits

Need for Rate Limiting

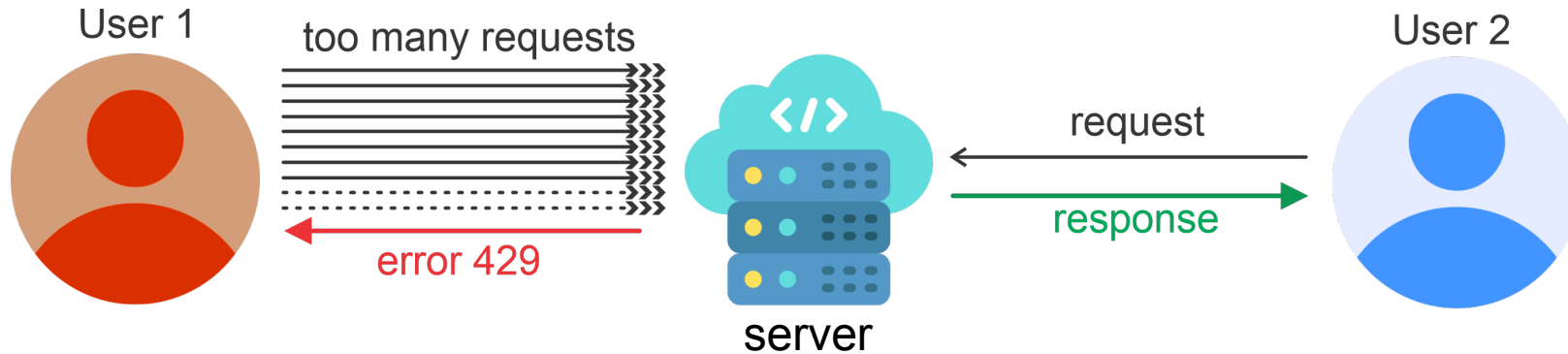


Denial of Service (DoS)

API abuse

Rate Limiting

Rate limiting is a strategy for limiting network traffic



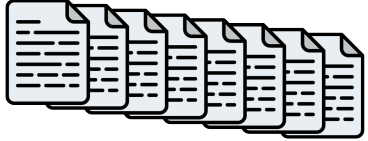
Why is it Important?

Prevents abuse

Ensures fair usage

Protects system from overload

Scaling Up: Large-scale text processing with Azure API

```
loop (   
  {  
    1. Read next file from disk  
    2. Call API  
  }
```

Be mindful of

Requests Per Minute (RPM)

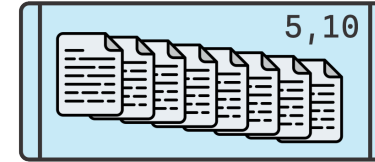
Tokens Per Minute (TPM)

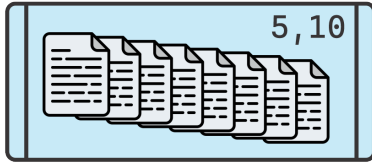
Scaling Up: Large-scale text processing with Azure API

1. Create queue



2. Put all files in queue



loop ()
{

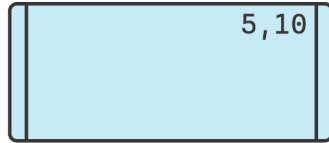
3. read next file from **queue**

4. call API

}

Scaling Up: Large-scale text processing with Azure API

1. Create queue



2. Put all files in queue



3. Create a pool of threads, run in parallel to process the queue

```
loop (  )  
{  
  _____  
  _____  
}
```

```
loop (  )  
{  
  _____  
  _____  
}
```

```
loop (  )  
{  
  _____  
  _____  
}
```

```
loop (  )  
{  
  _____  
  _____  
}
```

Scaling Up: Large-scale text processing with Azure API

❗ Error

Error code: 429 - {'error': {'code': '429', 'message': 'Requests to the ChatCompletions_Create Operation under Azure OpenAI API version 2024-05-01-preview have exceeded token rate limit of your current AIServices S0 pricing tier. Please retry after 60 seconds. Please contact Azure support service if you would like to further increase the default rate limit.'}}

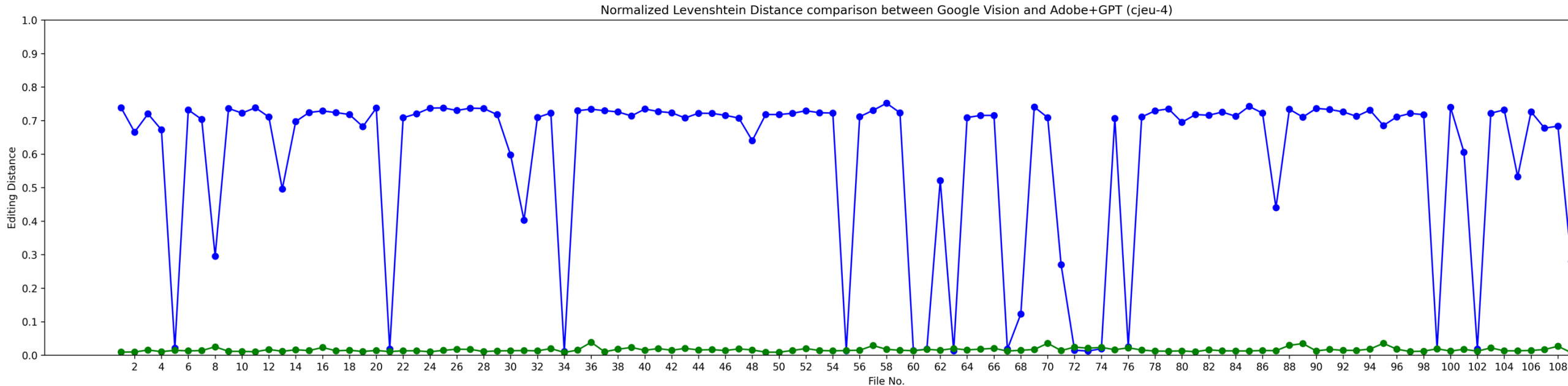
Why are we still encountering errors?

Tokens Per Minute (TPM) exceeded

How to address this problem?

Results on test data

- We had scans with manual correctionen
- Test prompts and models on this test set first
- «Error percentage» after Adobe OCR -> GPT-4 (green) looks good!
- Blue is Google Vision (not good at reading order)



Did we manage in the end?

- OCR scanning automated with Adobe's «Actions»
- GPT-4 averaged 110 s per page.
- API allows parallel calls (up to tpm quota): It took 7 days
- Due to time and cost, 30k already human-corrected pages were skipped

Points of contact at IT-Dep/dScience

- IT Department, general questions about API access etc: ffu-kontakt@usit.uio.no
- Knut's section: datafangst@usit.uio.no
- Sherin, dScience: contact@dscience.uio.no