

Calling OpenRouter large model aggregation service

Calling OpenRouter large model aggregation service

[Introduce](#)

[register](#)

[use](#)

[Common Errors](#)

1. Request error: -1

2. 401 No auth credentials found

3. Stuck at "Parsing non-streaming response - Status code: 200"

4. The reply is empty

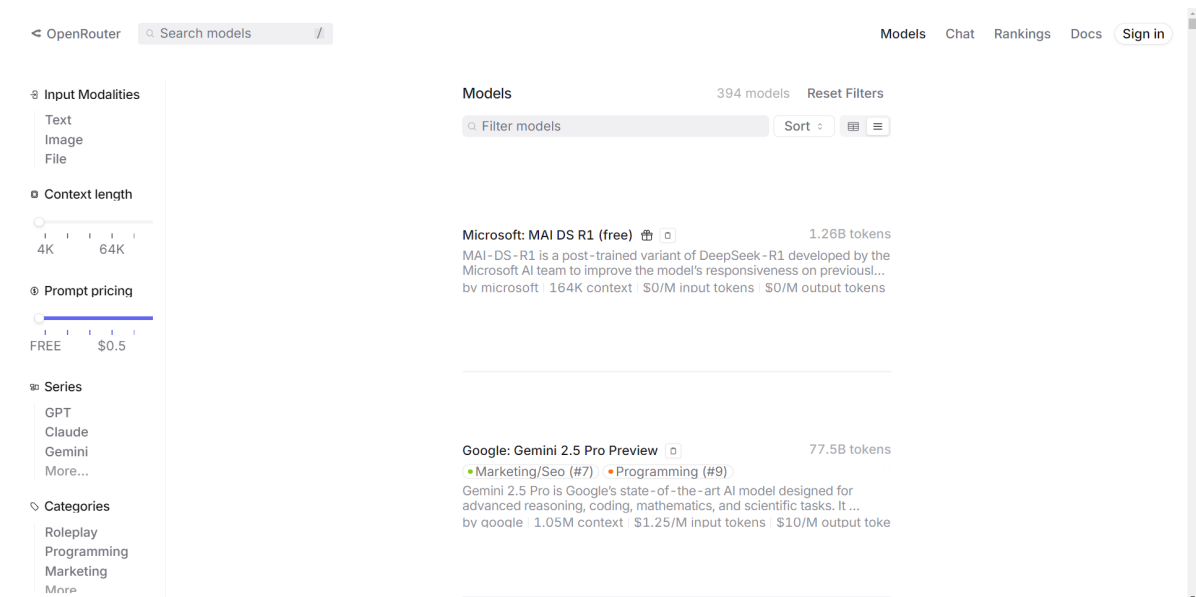
The list of currently available models is as follows:

Introduce

This tutorial has a high degree of operational difficulty

OpenRouter is a platform that aggregates a large number of mainstream AI language model APIs. Through the services provided by OpenRouter, we can easily switch/call various large models.

[Models](#) | [OpenRouter](#)

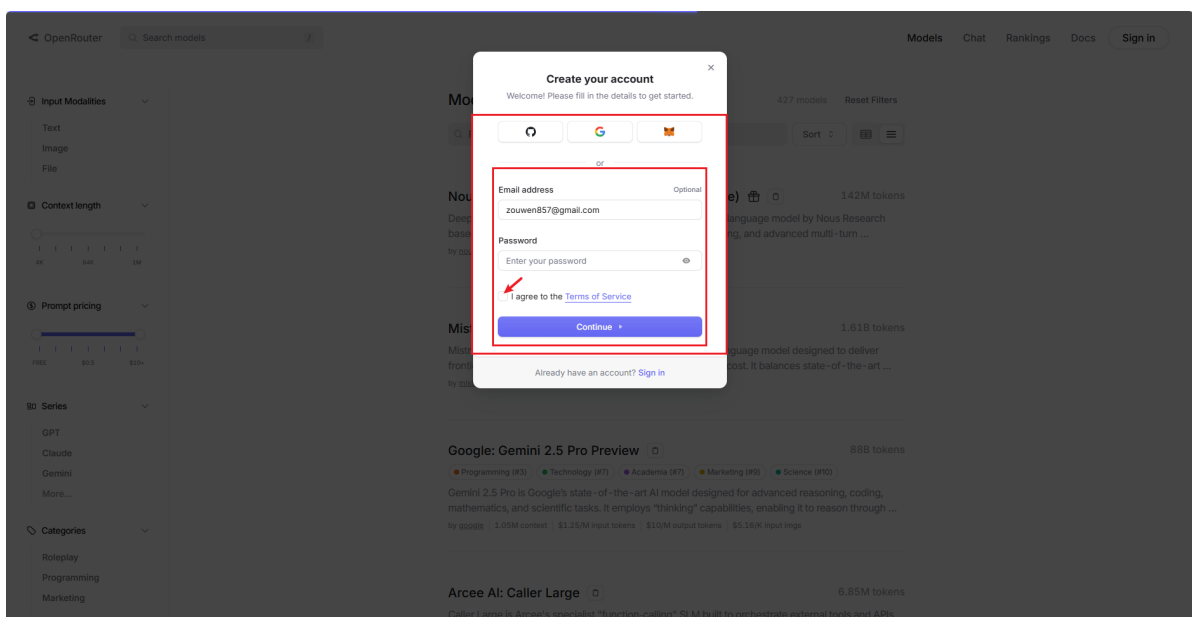
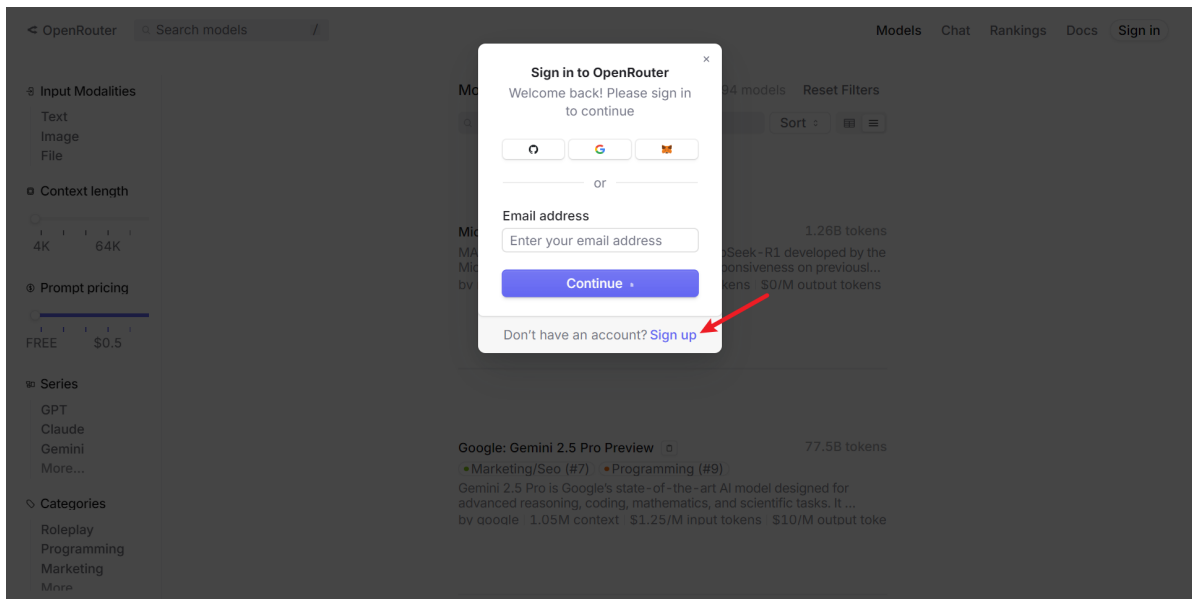
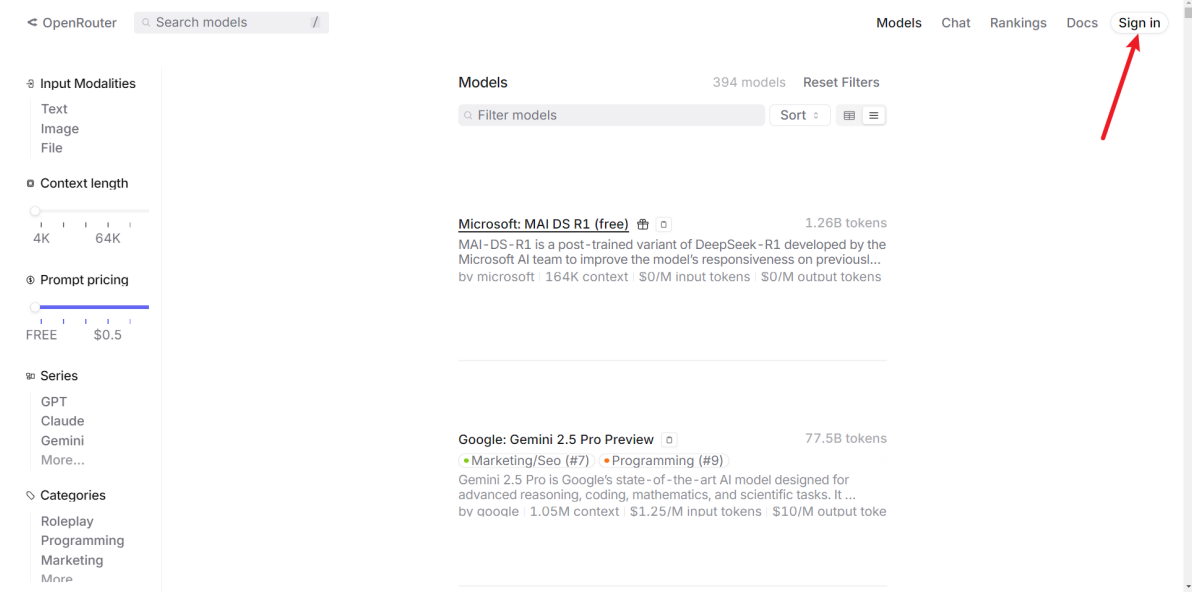


Due to the policies of various large model providers, some models on OpenRouter are inaccessible, which is normal.

The charges for different large models are also determined by the policies of the large model providers.

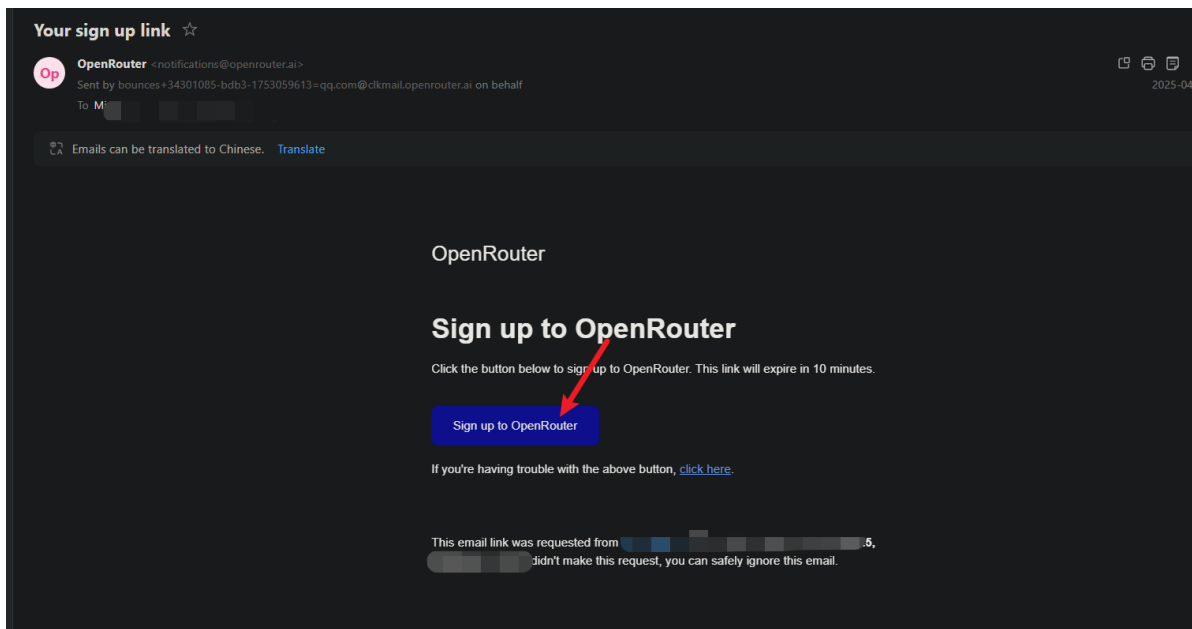
In most cases, the free models are sufficient for testing or simple use.

register

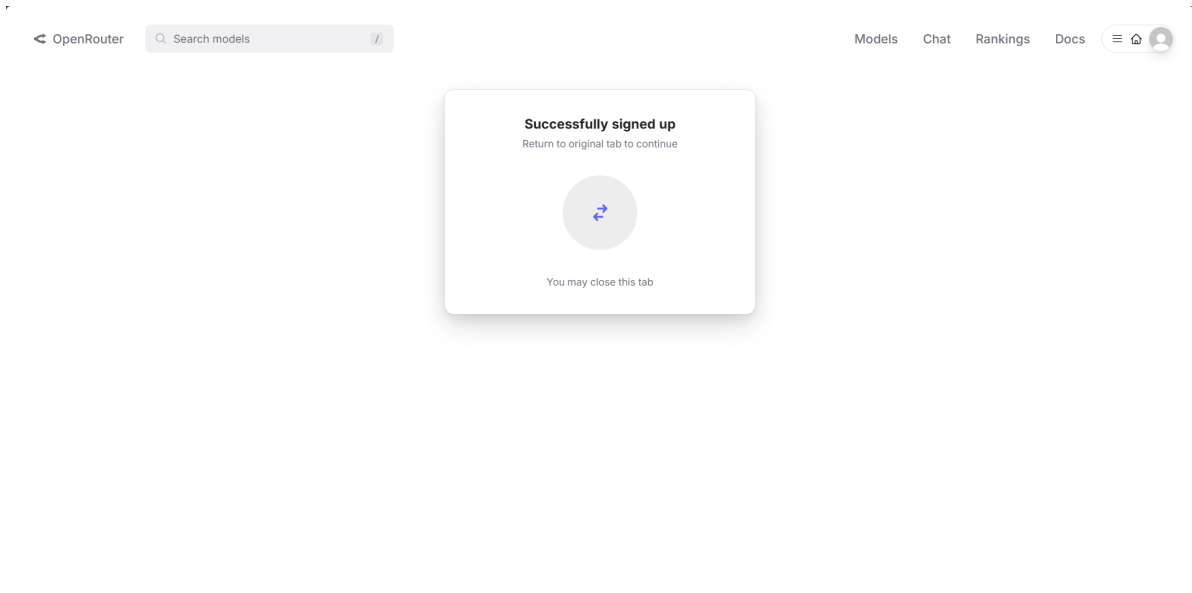


You can register using github / Google / MetaMask wallet or other email addresses

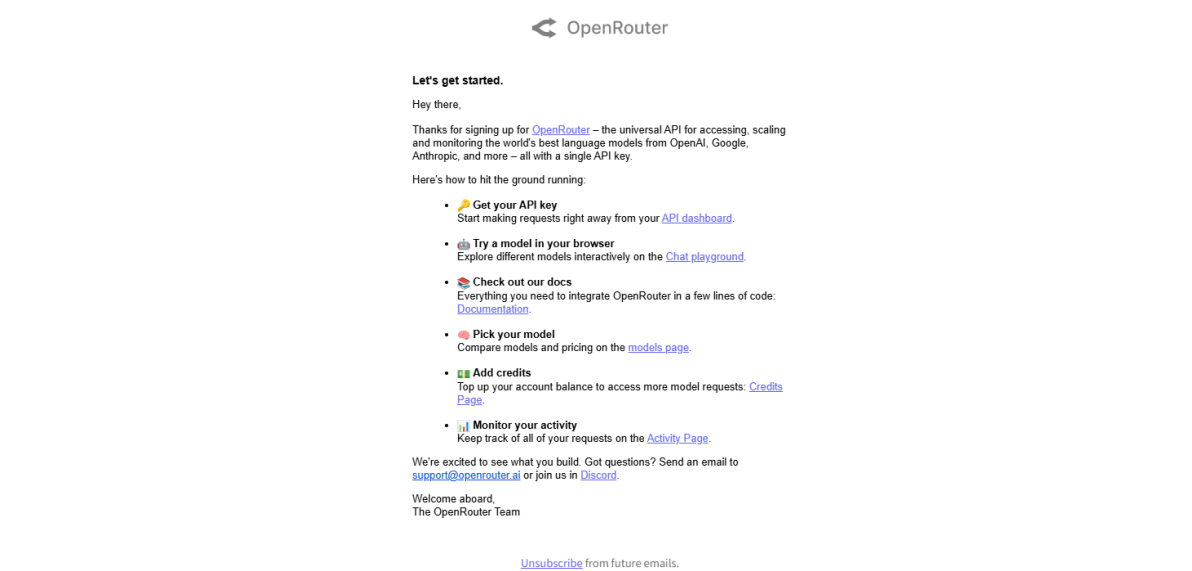
After clicking Continue, OpenRouter will send a verification link to your email.



Click this blue button



Or an email like this:





Click on the blue "API dashboard"

Let's get started.

Hey there,

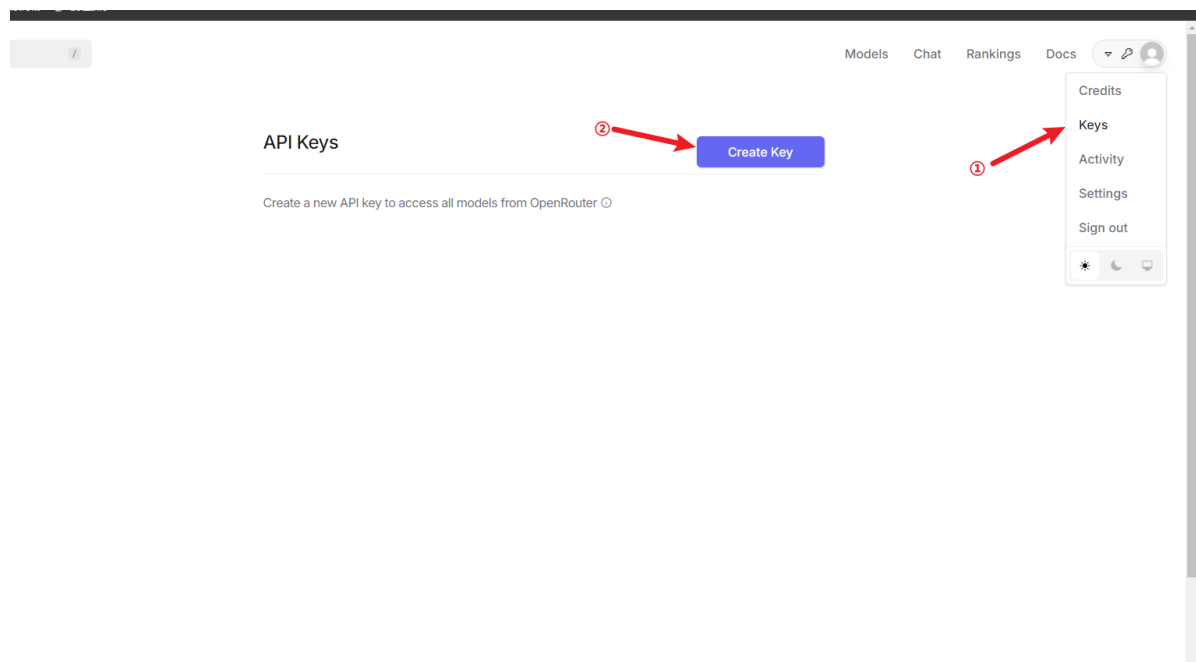
Thanks for signing up for [OpenRouter](#) – the universal API for accessing, scaling and monitoring the world's best language models from OpenAI, Google, Anthropic, and more – all with a single API key.

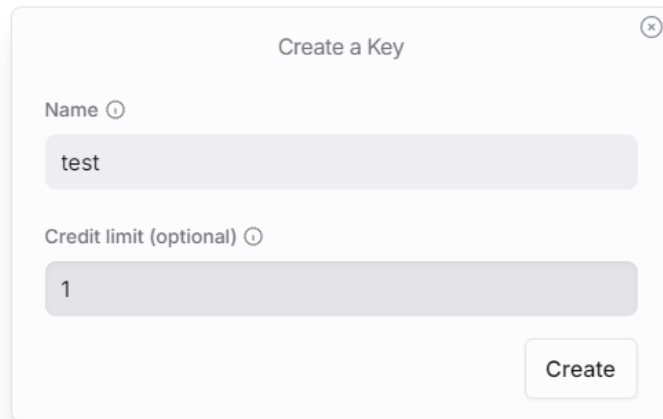
Here's how to hit the ground running:

-  **Get your API key**
Start making requests right away from your [API dashboard](#).
-  **Try a model in your browser**
Explore different models interactively on the [Chat playground](#).

This pop-up interface indicates that the registration is successful.

Next, let's create an API key



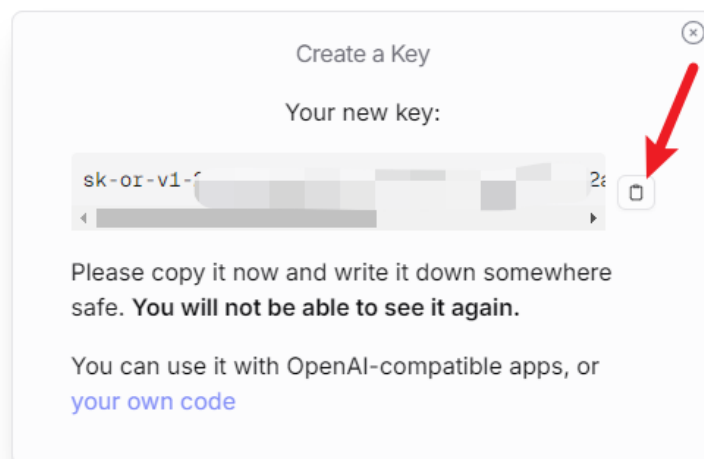


The first line is the API name, which can be filled in at will

The second line is the API consumption limit and can be left blank

This API Key will only be displayed once after it is generated. Please copy the API Key immediately and write it down.

Click the button on the right to copy

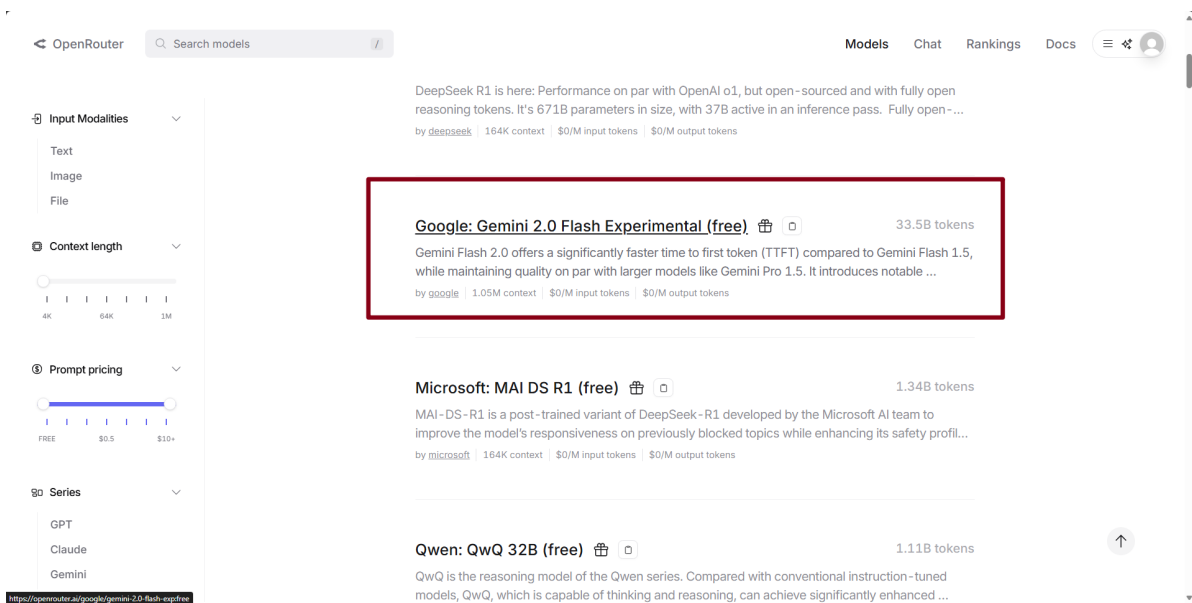
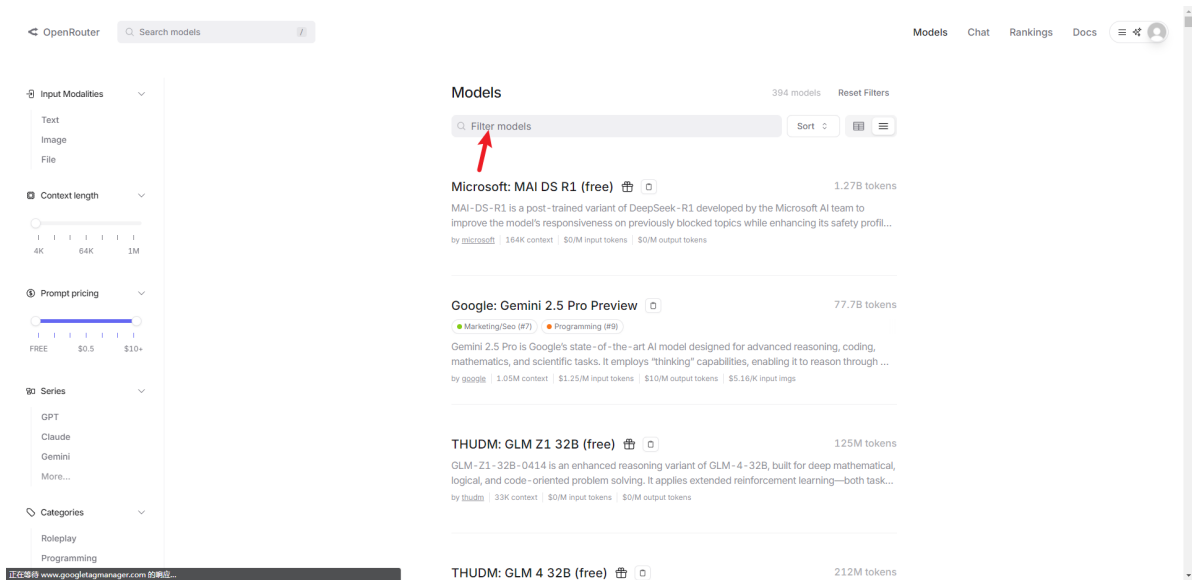


use

Let's randomly choose a model to test

<https://openrouter.ai/models>

Find the models page and enter free here



Let's take Google's Gemini 2.0 model as an example

Note: Some models may fail in actual searches. Here we can just find a test marked with free.

Clicking this model may cause a black screen or freeze in some cases, but it does not affect our use.

We just need to add a /api after the link and press Enter.

For example, the link in our browser now is:

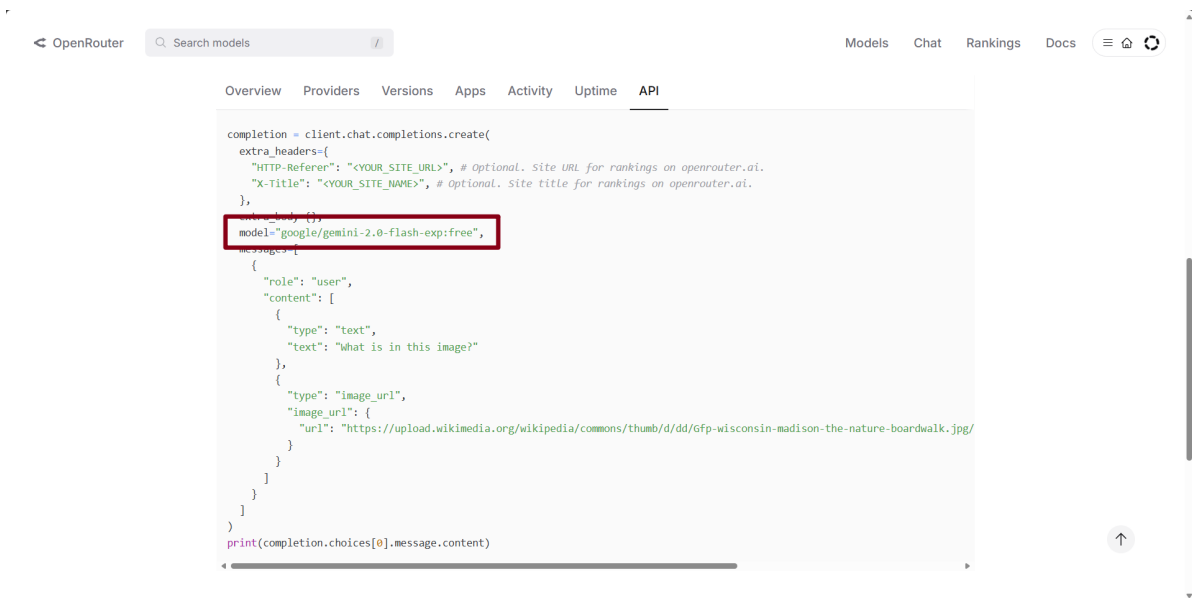
```
https://openrouter.ai/google/gemini-2.0-flash-exp:free
```

We need to change to

```
https://openrouter.ai/google/gemini-2.0-flash-exp:free/api
```

Then press Enter to access

Let's scroll down and find this paragraph



Copy this model name

google/gemini-2.0-flash-exp:free

Then we open [Source Code Summary / 11.Network / 08.LLM / llm_test.py]

```
from ybutils.LLM import *

def network_use_wlan(ssid, key):
    import network
    sta = network.WLAN(0)
    sta.connect(ssid, key)
    while not sta.isconnected():
        time.sleep(1)
    return sta.ifconfig()[0]

# 测试代码 Test code
if __name__ == "__main__":

    # 连接网络
    print("等待连接网络 waiting connect to wifi")
    network_use_wlan("SSID", "PASSWORD")

    # Spark示例
    spark_api_key = "API Password"

    print("正在等待响应 waiting for response ...")
    simple_chat_example(
        api_key=spark_api_key,
        prompt="sing me a song",
        model_type="openrouter",
        model="google/gemini-2.0-flash-exp:free"
    )
```

Fill in the WIFI name and password in the network_use_wlan method

The spark_api_key parameter is set to the API Key we just saved

Other parameter settings are as follows

```
model_type="openrouter",  
model="google/gemini-2.0-flash-exp:free"
```

Then we click Run

If you encounter the following error when running for the first time

```
[DEBUG] Request error: -1  
[DEBUG] Received full response: {'error': {'message': '\u8bf7\u6c42\u9519\u8bef: -1', 'type': 'request_error'}}  
{'error': {'message': '\u8bf7\u6c42\u9519\u8bef: -1', 'type': 'request_error'}}  
Error: {'message': '\u8bf7\u6c42\u9519\u8bef: -1', 'type': 'request_error'}
```

This is normal, we just need to run it again

Under normal circumstances, we can see the following information output:

```
[DEBUG] Request data length: 261  
[DEBUG] Creating socket connection to openrouter.ai:443  
[DEBUG] Resolved address: ('104.22.49.189', 443)  
[DEBUG] Using HTTPS connection  
[DEBUG] Received 1024 bytes  
[DEBUG] Received 1024 bytes  
[DEBUG] Received 185 bytes  
[DEBUG] Total response size: 2233 bytes  
[DEBUG] Response status code: 200  
[DEBUG] Closing connection  
[DEBUG] 收到响应 - 状态码 Received response - status code: 200  
[DEBUG] 尝试解析JSON响应Trying to parse JSON response  
[DEBUG] 处理非流式响应 Processing non-streaming response  
[DEBUG] 解析非流式响应 - 状态码 Parsing non-streaming response - status code: 200  
回复: (verse 1)  
The sun is shining, birds are singing free  
A gentle breeze is whispering through the tree  
The world is waking, str .....  
  
使用的token Tokens used: 310 (输入 input: 4, 输出 output: 306)
```

This means the interaction is successful!!

Common Errors

Calling the OpenRouter large model is a relatively complicated process. During the calling process, we are likely to encounter various problems.

Below we list some common abnormal situations

1. Request error: -1

```
[DEBUG] Request error: -1  
[DEBUG] Received full response: {'error': {'message': '\u8bf7\u6c42\u9519\u8bef: -1', 'type': 'request_error'}}  
{'error': {'message': '\u8bf7\u6c42\u9519\u8bef: -1', 'type': 'request_error'}}  
Error: {'message': '\u8bf7\u6c42\u9519\u8bef: -1', 'type': 'request_error'}
```


Occasional errors, more likely to occur when power is first turned on

Please rerun the program

2. 401 No auth credentials found

Please check if the API KEY is entered correctly.

```
[DEBUG] Handling non-streaming responses
[DEBUG] Parsing non-streaming response - Status code: 401
[DEBUG] Error response data: {'error': {'message': 'No auth credentials found',
'code': 401}}
[DEBUG] Received full response: {'error': {'message': 'No auth credentials
found', 'code': 401}}
{'error': {'message': 'No auth credentials found', 'code': 401}}
Error: {'message': 'No auth credentials found', 'code': 401}
```

3. Stuck at "Parsing non-streaming response - Status code: 200"

If the program gets stuck at this step, try changing the model

```
[DEBUG] Received response - Status code: 200
[DEBUG] Handling non-streaming responses
[DEBUG] Parsing non-streaming response - Status code: 200
```

4. The reply is empty

The reply is too long and has been truncated. Please try again.

The list of currently available models is as follows:

Most models are usable. I only tested a few of the most popular models.

```
GOOGLE:google/gemini-2.0-flash-exp:free
DEEPSEEK:deepseek/deepseek-r1:free
Microsoft: microsoft/mai-ds-r1:free
Alitongyiqianwen:qwen/qwq-32b:free
```