Using the Embedding API

The source code for this article is **here** . If you need to reproduce, you can download and run the source code.

1. Using OpenAI API

GPT has a packaged interface, we can simply package it. Currently, there are three GPT embedding modes, and the performance is as follows: |Model| Pages per dollar| **MTEB** score| **MIRACL** score| | --- | --- | --- | --- | |text-embedding-3-large|9,615|54.9|64.6| |text-embedding-3-small|62,500|62.3|44.0| |text-embedding-ada-002|12,500|61.0|31.4|

- The MTEB score is the average score of eight tasks including embedding model classification, clustering, and pairing.
- The MIRACL score is the average score of the embedding model on the retrieval task.

From the above three embedding models, we can see `text-embedding-3-large` that the best performance and the most expensive price are available. We can use it when the application we build requires better performance and the cost is sufficient; the best `text-embedding-3-small` performance and price are available. We can choose this model when our budget is limited; the previous generation model of OpenAI is inferior to the previous two in terms of performance or price, so it is not recommended. `text-embedding-ada-002`

python

```python
import os
from openai import OpenAI
from dotenv import load_dotenv, find_dotenv


# 读取本地/项目的环境变量。
# find_dotenv()寻找并定位.env文件的路径
# load_dotenv()读取该.env文件，并将其中的环境变量加载到当前的运行环境中
# 如果你设置的是全局的环境变量，这行代码则没有任何作用。
_ = load_dotenv(find_dotenv())

# 如果你需要通过代理端口访问，你需要如下配置
```

```python
os.environ['HTTPS_PROXY'] = 'http://127.0.0.1:7890'
os.environ["HTTP_PROXY"] = 'http://127.0.0.1:7890'

def openai_embedding(text: str, model: str=None):
    # 获取环境变量 OPENAI_API_KEY
    api_key=os.environ['OPENAI_API_KEY']
    client = OpenAI(api_key=api_key)

    # embedding model: 'text-embedding-3-small', 'text-embedding-3-large',
    if model == None:
        model="text-embedding-3-small"

    response = client.embeddings.create(
        input=text,
        model=model
    )
    return response

response = openai_embedding(text='要生成 embedding 的输入文本，字符串形式。')
```

The data returned by the API is in `json` the format of vector type, which `object` also includes the data stored `data`, embedding model type `model`, and token usage `usage`, as shown below:

```json
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [
        -0.006929283495992422,
        ... (省略)
        -4.547132266452536e-05,
      ],
    }
  ],
  "model": "text-embedding-3-small",
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 5
```

```
  }
}
```

We can call the response object to get the embedding type.

python

```
print(f'返回的embedding类型为: {response.object}')
```

markup

```
返回的embedding类型为: list
```

The embedding is stored in data, and we can check the length of the embedding and the generated embedding.

python

```
print(f'embedding长度为: {len(response.data[0].embedding)}')
print(f'embedding (前10) 为: {response.data[0].embedding[:10]}')
```

markup

```
embedding长度为: 1536
embedding (前10) 为: [0.03884002938866615, 0.013516489416360855, -0.0024250
```

We can also view the model and token usage of this embedding.

python

```
print(f'本次embedding model为: {response.model}')
print(f'本次token使用情况为: {response.usage}')
```

markup

```
本次embedding model为: text-embedding-3-small
本次token使用情况为: Usage(prompt_tokens=12, total_tokens=12)
```

2. Using Wenxin Qianfan API

Embedding-V1 is a text representation model based on Baidu Wenxin large model technology. Access token is the credential for calling the interface. When using Embedding-V1, you should first obtain Access token with API Key and Secret Key, and then use Access token to call the interface to embed text. At the same time, Qianfan large model platform also supports embedding models such as bge-large-zh.

python

```python
import requests
import json

def wenxin_embedding(text: str):
    # 获取环境变量 wenxin_api_key、wenxin_secret_key
    api_key = os.environ['QIANFAN_AK']
    secret_key = os.environ['QIANFAN_SK']

    # 使用API Key、Secret Key向https://aip.baidubce.com/oauth/2.0/token 获取
    url = "https://aip.baidubce.com/oauth/2.0/token?grant_type=client_cre
    payload = json.dumps("")
    headers = {
        'Content-Type': 'application/json',
        'Accept': 'application/json'
    }
    response = requests.request("POST", url, headers=headers, data=payloa

    # 通过获取的Access token 来embedding text
    url = "https://aip.baidubce.com/rpc/2.0/ai_custom/v1/wenxinworkshop/e
    input = []
    input.append(text)
    payload = json.dumps({
        "input": input
    })
    headers = {
        'Content-Type': 'application/json'
    }

    response = requests.request("POST", url, headers=headers, data=payloa

    return json.loads(response.text)
# text应为List(string)
text = "要生成 embedding 的输入文本，字符串形式。"
response = wenxin_embedding(text=text)
```

In Embedding-V1, each embedding has a separate id and a timestamp to record the time of embedding.

```python
print('本次embedding id为：{}'.format(response['id']))
print('本次embedding产生时间戳为：{}'.format(response['created']))
```

```markup
本次embedding id为：as-hvbgfuk29u
本次embedding产生时间戳为：1711435238
```

Similarly, we can also get the embedding type and embedding from the response.

```python
print('返回的embedding类型为:{}'.format(response['object']))
print('embedding长度为：{}'.format(len(response['data'][0]['embedding'])))
print('embedding（前10）为：{}'.format(response['data'][0]['embedding'][:10
```

```markup
返回的embedding类型为:embedding_list
embedding长度为：384
embedding（前10）为：[0.060567744076251984, 0.020958080887794495, 0.0532342
```

3. Use iFlytek Spark API

Not yet open

4. Using Zhipu API

Zhipu has a packaged SDK that we can call.

```python
from zhipuai import ZhipuAI
def zhipu_embedding(text: str):
```

```
        api_key = os.environ['ZHIPUAI_API_KEY']
        client = ZhipuAI(api_key=api_key)
        response = client.embeddings.create(
            model="embedding-2",
            input=text,
        )
        return response

    text = '要生成 embedding 的输入文本，字符串形式。'
    response = zhipu_embedding(text=text)
```

response is of `zhipuai.types.embeddings.EmbeddingsResponded` type. We can call `object`, `data`, `model`, `usage` to view the embedding type, embedding, embedding model, and usage of the response.

python

```
print(f'response类型为：{type(response)}')
print(f'embedding类型为：{response.object}')
print(f'生成embedding的model为：{response.model}')
print(f'生成的embedding长度为：{len(response.data[0].embedding)}')
print(f'embedding（前10）为：{response.data[0].embedding[:10]}')
```

markup

```
response类型为：<class 'zhipuai.types.embeddings.EmbeddingsResponded'>
embedding类型为：list
生成embedding的model为：embedding-2
生成的embedding长度为：1024
embedding（前10）为：[0.017892399802803993, 0.0644201710820198, -0.0093428
```

The source code for this article is **here** . If you need to reproduce, you can download and run the source code.

Previous chapter
1. Introduction to word vectors and vector knowledge base

3. Data processing