

OpenAI GPT-4 ChatGPT LangChain 人口知能プログラミング実践入門

PDF版ダウンロードリンク

<https://www.borndigital.co.jp/book/30237.html>

Python3.10

LlamaIndex0.6.12

LangChain0.0.181

目的

テキストや画像生成はAPIを使ってプログラムから利用してみる

事前学習済モデルに追加学習を行わせる「ファインチューニング」の概要と実際の使用例を学ぶ

音声ファイルのテキスト変換機能や、OpenAIのポリシーに準拠しているかを確認する「モデレーション」を使ってみる

OpenAI API (Application Programming Interface)

OpenAI APIは、OpenAIが提供する自然言語処理（NLP）に特化したクラウドベースのAPI（Application Programming Interface）です。これにより、開発者や企業はOpenAIの強力な言語モデルであるGPT-3を自分のアプリケーションやプロダクトに統合できます。

GPT-3は、大規模なトレーニングデータと深層学習技術を活用して、非常に高度な自然言語理解と生成を行えるAIモデルです。

サポートする機能・タスクリスト

文章の生成

質問応答

文章の要約

翻訳

自動作文

言語に関連するタスク

多岐にわたるNLPタスク

APIを使用する際は、OpenAIのウェブサイトでAPIキーを取得し、それを利用してAPIエンドポイントにリクエストを送信します。APIはRESTfulな形式で提供されており、Pythonや他のプログラミング言語を使用して簡単に統合することができます。

また、OpenAI APIは「プロダクションAPI」と「研究API」の2つのモードがあります。プロダクションAPIは商用アプリケーションに適しており、利用には料金が発生します。一方、研究APIは非商用目的での利用や試験などに無料でアクセスできます。

ただし、OpenAI APIには使用制限やガイドラインが存在し、適切な利用を守る必要があります。公正な使用と倫理的な観点を重視するよう心掛けることが重要です。

APIキーの取得

APIキー取得の流れ:

1. ホームページへアクセス

<https://openai.com/api/> にアクセスし「LOG IN」をクリックしてログインします。ユーザー名とパスワードはChatGPTを利用するため作成したOpenAIのアカウントのものを使用してください。

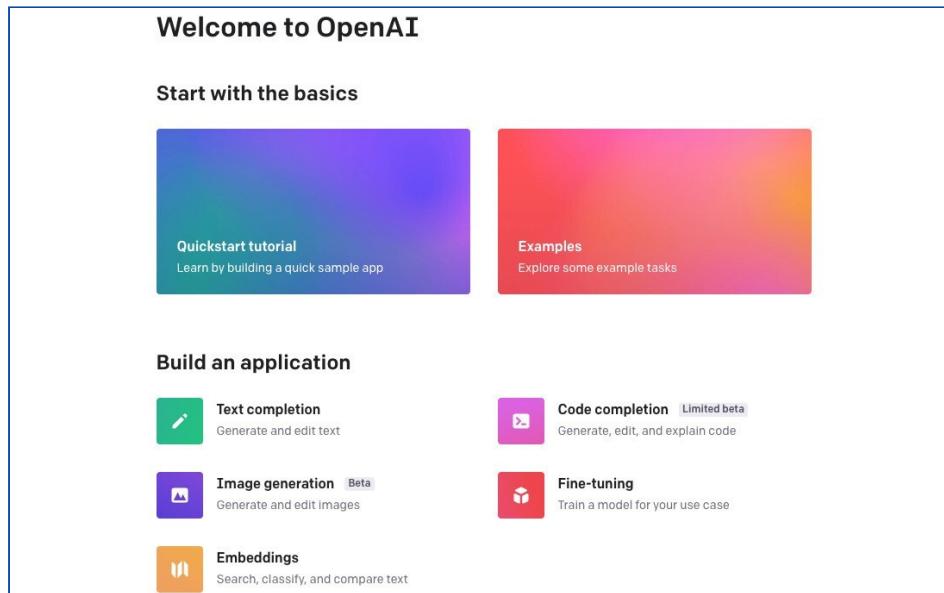


図4.1 出所: ソフトアンテナより. 公式Webサイト |OpenAIのホームページ

2. 右上のアイコンをクリックし、「View API keys」メニューを選択する

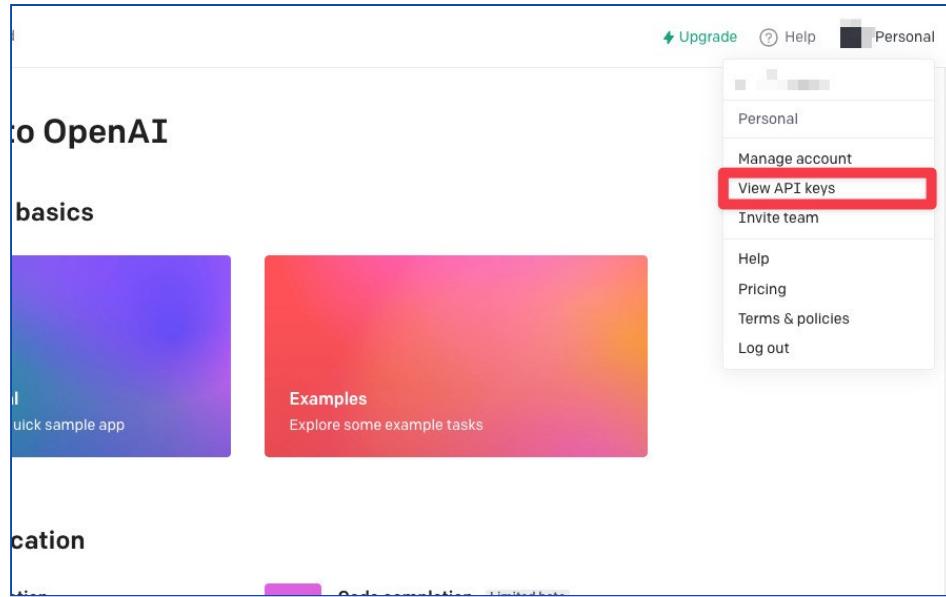


図4.2 出所: ソフトアンテナより. 公式Webサイト |OpenAIのアカウント画面

3. 「Create new secret key」ボタンをクリックする

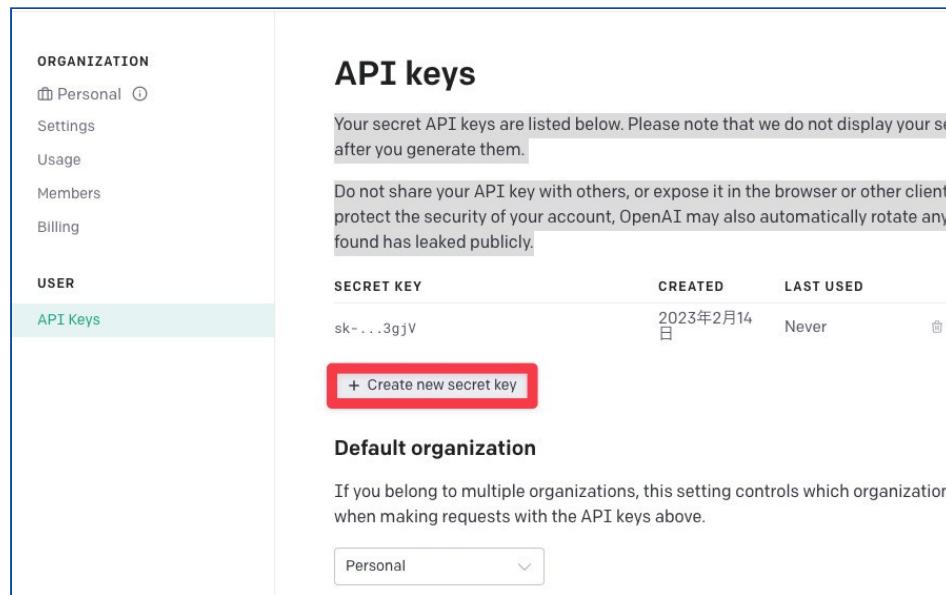


図4.3 出所: ソフトアンテナより. 公式Webサイト |OpenAIのアカウントのAPIキー取得ボタン画面

4. APIキーが表示され、生成されたAPIキーは再表示することができないため、安全に保存しておく必要がある

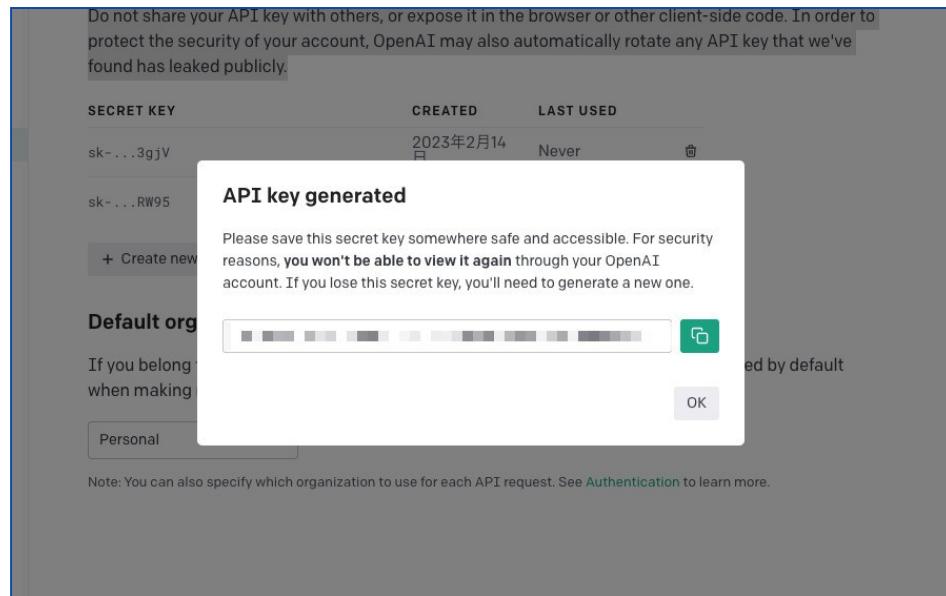


図4.4 出所: ソフトアンテナより. 公式Webサイト |APIキーの表示画面

5. キーの取得完了

OpenAI APIのライブラリ

OpenAI APIのライブラリがサポートする言語:

Python

Node.js

その他多くの言語をサポートしているコミュニティがある

詳細については以下のリンクにてご参照ください

<https://platform.openai.com/docs/libraries>

テキスト生成について

公式サイト

テキスト生成関連

<https://platform.openai.com/docs/guides/completion>

チャート生成関連

<https://platform.openai.com/docs/guides/chat>

end-user ID-OpenAI API

<https://platform.openai.com/docs/guides/safety-best-practices/end-user-ids>

テキスト生成には4つの機能がある:

- テキスト生成
- チャート
- 挿入
- 編集

利用料金:

OpenAI Playgroundの利用料金と同じである

常に実行するコードセットを用意し、利用パラメーターも料金プラン※範囲内に入るよう設定しておく必要がある

設定パラメーター料金プランはOpenAI Playgroundの利用料金・トークン数についてをご参照ください

尚) 各自「生成に要するパラメーター」は決まっているので各自のパラメーター基準表に従ってください

`openai.Completion.create()`:テキスト生成パラメーター

`openai.ChatCompletion.create()`:チャート生成パラメーター

`openai.Edit.create()`:テキス編集パラメーター

尚) 各自生成出力に要する「レスポンスパラメーター」は決まっているので各自のパラメーター基準表に従ってください (*ユーザー入力がある場合)

`openai.Completion.create()`:テキスト生成のレスポンスパラメーター

`openai.ChatCompletion.create()`:チャート生成のレスポンスパラメーター

`openai.Edit.create()`:テキス編集のレスポンスパラメーター

前準備:

1. Colabノートブックを開く

2. openaiパッケージをインストールする

`!pip install openai`

3. 環境変数の準備をする(有料版、無料版の場合は不要)

```
import os
os.environ["OPENAI_API_KEY"] = "<OpenAI_APIのAPIキー>"
```

入力(プロンプト)の準備:

1. 予めに「`prompt = 入力する内容`」を以下のように書いて準備しておく必要がある

物語を書いてください
コミュ障でぼっちな学生があることをきっかけにロックバンドに加入し、慣れない人間関係を通じて活動していく物語

2. テキスト生成に要するパラメーター例:

```
import openai

api_key = "YOUR_API_KEY"
openai.api_key = api_key

response = openai.Completion.create(
    engine="davinci",
    prompt="Once upon a time",
    temperature=0.7,
    max_tokens=100,
```

ここから以後は高度な設定が必要な場合のみ使用

```
    stop=["\n"],
    n=1,
    user="user123",
    frequency_penalty=0.2,
    presence_penalty=0.5,
    log_level="info",
    logprobs=0,
    echo=False
)

print(response['choices'][0]['text'])
```

3. テキスト生成出力に要するレスポンスパラメーター例:

```
import openai

api_key = "YOUR_API_KEY"
openai.api_key = api_key

response = openai.Completion.create(
    engine="davinci",
    prompt="こんにちは、私はChatGPTです。",
    max_tokens=100,
    temperature=0.7,
    stop=["\n"],
    api_key=api_key,
)

generated_text = response['choices'][0]['text']
print(generated_text)
```

openai.Completion.create()の作成パラメーターの説明

パラメーターの解説:

1. パラメーターを指定

```
openai.Completion.create(
```

2. 使用するGPT-3エンジンを指定

```
    engine="davinci",
```

3. テキストの入力プロンプトを指定

```
    prompt="こんにちは、私はChatGPTです。",
```

4. 出力されるテキストの最大トークン数を指定

※トークン数は最大4096まで※

```
    max_tokens=100,
```

5. 出力の多様性を制御するための温度パラメーターを指定

※デフォルト: 1、0～2範囲内に設定※

```
    temperature=0.7,
```

6. 応答の終了条件を指定

※生成を停止する回数は最大4回まで※

```
    stop=["\n"],
```

7. 既存のトークンに対する再使用のペナルティを指定

※デフォルト: 0、-2～2範囲内に設定※

```
    frequency_penalty=0.2,
```

8. 特定のトークンを出力に含めるためのペナルティを指定

※デフォルト: 0、-2～2範囲内に設定※

```
    presence_penalty=0.5,
```

9. 生成する応答の数を指定

```
n=1,
```

オプション:ユーザーIDなどの識別情報を指定

※これはエンドユーザーのIDであり、実行するIDではない※

```
user="user123",
```

10.

オプション：対話型の応答を示す例を定義

```
examples=[  
    ["こんにちは", "こんにちは!"],  
    ["お元気ですか?", "はい、元気です!"],  
],  
)
```

11.

※実行結果省略:APIキー入力がないと観覧できないため※

```
In [ ]: #openaiパッケージをインストールする  
!pip install openai
```

```
Collecting openai
  Downloading openai-0.27.9-py3-none-any.whl (75 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.5/75.5 kB 1.4 MB/s eta 0:00:00
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from openai) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from openai) (3.8.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2023.7.22)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (4.0.3)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.3.1)
Installing collected packages: openai
Successfully installed openai-0.27.9
```

```
In [ ]: #テキスト生成(無料版)
#promptを用意する

prompt = '''以下の物語を書いてください
ギターを愛するもののコミュニティでぼっちはな学生があることをきっかけにロックバンドに加入し、慣れない人間同士の繋がりを発展させる物語を書く。'''

# テキスト生成
```

```
In [ ]: import openai

# GPT-3モデルを指定してテキスト生成をする
response = openai.Completion.create(
    model="text-davinci-003",
    prompt=prompt,
    max_tokens=500,
    temperature=0.7
)

# 生成されたテキストを表示する
print(response['choices'][0]['text'])

# 丁度500トークン数に区切るため文章が終わらない場合がある(↓参照出力例文)

#。
# 学生の春夜はギターを愛していた。自分一人の世界に籠りこんでいた彼女は、音楽以外に何をするかと悩んでいた。
# そんなある日、春夜は音楽部の労働祭でロックバンドを結成したと聞いた。彼女にとって、これが夢のよう
```

```
# 春夜は焦りながらも、ギターを抱えて音楽部に入った。他のメンバーには一人一人が今までにない楽器を持っていた。
# 春夜は彼らの熱意に圧倒され、最初は恐れながらも活動に参加していった。だがその後、仲間たちとの交流を通じて、春夜も自信を取り戻していった。
# 彼女も他のメンバーと同じように、新しい音楽の創造に取り組むようになった。彼女の努力は報われ、彼女が創り上げた曲は学校の伝説となった。
# 最終的に、春
```

。

学生の春夜はギターを愛していた。自分一人の世界に籠りこんでいた彼女は、音楽以外に何をするかというと、何もできないコミュ障だった。

そんなある日、春夜は音楽部の労働祭でロックバンドを結成したと聞いた。彼女にとって、これが夢のようなチャンスだった。

春夜は焦りながらも、ギターを抱えて音楽部に入った。他のメンバーには一人一人が今までにない楽器を持っていて、それぞれの色を混ぜて新しい音楽を創り上げようとしていた。

春夜は彼らの熱意に圧倒され、最初は恐れながらも活動に参加していった。だがその後、仲間たちとの交流を通じて、春夜も自信を取り戻していった。

彼女も他のメンバーと同じように、新しい音楽の創造に取り組むようになった。彼女の努力は報われ、彼女が創り上げた曲は学校の伝説となった。

最終的に、春

```
In [ ]: #テキスト生成の例(有料APIキー仕様版)
import openai

# OpenAI APIを使用するためにAPIキーを設定する
api_key = "Set your OpenAI API key" # ここに実際のAPIキーを入力してください
openai.api_key = api_key

# GPT-3エンジンを指定してテキスト生成のAPIリクエストを作成する
response = openai.Completion.create(
    engine="davinci",
    prompt="こんにちは、私はChatGPTです。",
    max_tokens=100,
    temperature=0.7,
    stop=["\n"],
    api_key=api_key,
)

# APIリクエストの結果から生成されたテキストを取得する
generated_text = response['choices'][0]['text']

# 生成されたテキストをコンソールに表示する
print(generated_text)
#これは私の最近です。とGPTのAIが返してくれる。
```

これは私の最近です。

openai.Completion.create()の出力・レスポンスパラメーターの説明

エンドユーザーなどがある場合、レスポンスの定義パラメーターをプログラムする必要がある

リクエストでエンドユーザーIDを指定することで、OpenAIアプリケーションでポリシー違反が検出された場合に、より実用的なフィードバックをチームに提供できる

End-user ID-OpenAI API <https://platform.openai.com/docs/guides/safety-best-practices/end-user-ids>

レスポンスパラメーターの設定は以下の例の通りにプログラムを書く

レスポンスパラメーターの解説：

1. openai環境準備をする

```
import openai
```

2. OpenAI APIを使用するためにAPIキー(有料) を設定

" "ここに実際のAPIキーを入力してください

```
api_key = "YOUR_API_KEY"
```

```
openai.api_key = api_key
```

3. GPT-3エンジンを指定してテキスト生成のAPIリクエストを作成

```
response = openai.Completion.create(
```

4. 使用するGPT-3エンジンを指定

```
    engine="davinci",
```

5. テキストの入力プロンプトを指定

```
    prompt="こんにちは、私はChatGPTです。",
```

6. 出力されるテキストの最大トークン数を指定

```
    max_tokens=100,
```

7. 出力の多様性を制御するための温度パラメーターを指定

```
    temperature=0.7,
```

8. 応答の終了条件を指定

```
    stop=["\n"],
```

```
    api_key=api_key,  
)
```

9. APIリクエストの結果から生成されたテキストを取得

```
generated_text = response['choices'][0]['text']
```

```
print(generated_text)
```

- 10.

※実行結果省略:APIキー入力がないと観覧できないため※

```
In [ ]: #顧客ユーザーID指定があるときのテキスト生成の例
import openai

api_key = "YOUR_API_KEY" # Replace this with your actual API key
openai.api_key = api_key

response = openai.Completion.create(
    engine="davinci",
    prompt="Once upon a time",
    temperature=0.7,
    max_tokens=100,
    stop=["\n"],
    n=1,
    user="user123",
    frequency_penalty=0.2,
    presence_penalty=0.5,
    log_level="info",
    logprobs=0,
    echo=False
)

print(response['choices'][0]['text'])
```

質問応答のテキスト生成

テキスト生成と同様に設定するが、入力する内容は「文章」から「質問？」に変わった
だけで他はやることが一緒なので「テキスト生成の例をご参照ください」

```
In [ ]: #質問応答テキスト生成
#promptを用意する

prompt = "人工知能について教えてください"
```

```
In [ ]: import openai

# GPT-3モデルを指定してテキスト生成をする
response = openai.Completion.create(
    model="text-davinci-003",
    prompt=prompt,
    max_tokens=500,
    temperature=0.7
)

# 生成されたテキストを表示する
print(response['choices'][0]['text'])

#表示結果の例↓
#人工知能(AI)とは、コンピューターを用いて人間と同様に論理的思考を行うことを可能にする技術です。:
```

人工知能(AI)とは、コンピューターを用いて人間と同様に論理的思考を行うことを可能にする技術です。特に、パターン認識や自然言語処理、知識の再構築などの領域において、コンピューターが自動的かつ効率的に行動を行うことを目的としています。AIの使われる領域は多岐にわたり、自動運転車、ゲーム、銀行業務、画像処理などで使われています。

要約のテキスト生成

テキスト生成と同様に設定するが、入力する内容は「文章」から「要約してください」という指示を足すだけでOK」に変わっただけで他はやることが一緒なので「テキスト生成をご参照ください」

```
In [ ]: #要約テキスト生成  
#promptを用意する  
  
prompt = '''以下の文章を短い1文で要約してください  
  
人工知能(AI)とは、コンピューターを用いて人間と同様に論理的思考を行うことを可能にする技術です。特
```

```
In [ ]: import openai  
  
# GPT-3モデルを指定してテキスト生成をする  
response = openai.Completion.create(  
    model="text-davinci-003",  
    prompt=prompt,  
    max_tokens=500,  
    temperature=0.7  
)  
  
# 生成されたテキストを表示する  
print(response['choices'][0]['text'])  
  
#表示結果の例↓  
#人工知能は、パターン認識や自然言語処理、知識の再構築などを可能にした技術で、多岐にわたる領域
```

人工知能は、パターン認識や自然言語処理、知識の再構築などを可能にした技術で、多岐にわたる領域（自動運転車、ゲーム、銀行業務、画像処理など）で使用されている。

翻訳のテキスト生成

テキスト生成と同様に設定するが、入力する内容は「文章」から「翻訳してください」という指示を足すだけでOK」に変わっただけで他はやることが一緒なので「テキスト生成をご参照ください」

ただし、記号については細かく記載通りに指定する必要がある

翻訳のテキスト生成流れ:

翻訳部以外のコード解釈はテキスト生成と同じなのでそちらへご参照ください

```
!pip install openai  
import openai  
  
openai.api_key = "Set your OpenAI API key"
```

1. 翻訳したい文章を""ここに入力する

翻訳命令コード（英文例）

Translate the following 「元の言語」 text to 「翻訳後の言語」 :

```
prompt = "Translate the following English text to French:  
'Hello, how are you?'"
```

2. パラメーター設定

```
response = openai.Completion.create( engine="text-davinci-003", # Use GPT-  
3.5 engine temperature=0.7, prompt=prompt, max_tokens=100  
)
```

3. 表示定義

```
generated_text = response.choices[0].text.strip()
```

4. 表示命令

```
print("Generated Text:")  
print(generated_text)
```

※実行結果省略:APIキー入力がないと観覧できないため※

5. 出力

Generated Text:

Bonjour, comment allez-vous ?

```
In [ ]: #翻訳テキスト生成  
#promptを用意する  
  
prompt = '''日本語を英語に翻訳する。  
  
日本語:人工知能はパターン認識や自然言語処理技術である。  
英語:'''
```

```
In [ ]: import openai  
  
# GPT-3モデルを指定してテキスト生成をする  
response = openai.Completion.create(  
    model="text-davinci-003",  
    prompt=prompt,  
    temperature=0.7  
)  
  
# 生成されたテキストを表示する  
print(response['choices'][0]['text'])  
  
#表示結果の例  
#Artificial intelligence is a pattern recognition and natural language pr
```

Artificial intelligence is a pattern recognition and natural language processing technology.

```
In [ ]: #API有料版翻訳テキスト生成の例  
!pip install openai
```

```

import openai

openai.api_key = "Set your OpenAI API key"

prompt = "Translate the following English text to French: 'Hello, how are you?'"

response = openai.Completion.create(
    engine="text-davinci-003", # Use GPT-3.5 engine
    temperature=0.7,
    prompt=prompt,
    max_tokens=100
)

generated_text = response.choices[0].text.strip()

print("Generated Text:")
print(generated_text)

```

Collecting openai
 Downloading openai-0.27.8-py3-none-any.whl (73 kB)

73.6/73.6 kB 1.8 MB/s eta 0:00

Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from openai) (2.27.1)
 Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai) (4.65.0)
 Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from openai) (3.8.5)
 Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (1.26.16)
 Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2023.7.22)
 Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2.0.12)
 Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.4)
 Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (23.1.0)
 Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (6.0.4)
 Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (4.0.2)
 Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.9.2)
 Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.4.0)
 Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.3.1)
 Installing collected packages: openai
 Successfully installed openai-0.27.8
 Generated Text:
 Bonjour, comment allez-vous ?

プログラム生成

テキスト生成と同様に設定するが、入力する内容は「文章」から「プログラム内容を書いてください」という指示を足すだけでOK」に変わっただけで他はやることが一緒なので「テキスト生成をご参考ください」

ただし、モデルエンジンは「davinci」に変わる

プログラム生成部以外のコード解釈はテキスト生成と同じなのでそちらへご参照ください

```
import openai
```

1. OpenAI APIを使用するためにAPIキーを設定

```
" "ここに実際のAPIキーを入力してください
```

```
api_key = "YOUR_API_KEY"
openai.api_key = api_key
```

2. GPT-3エンジンを指定してテキスト生成のAPIリクエストを作成

```
response = openai.Completion.create( engine="davinci",
```

3. プログラム生成命令コード:

「Pythonでプログラミングするためのコードを教えてください。」

```
prompt="Pythonでプログラミングするためのコードを教えてください。",
max_tokens=100,
temperature=0.7,
stop=["\n"],
api_key=api_key,
)
```

4. APIリクエストの結果から生成されたテキストを取得 generated_text =

```
response['choices'][0]['text']
```

5. 生成されたテキストをコンソールに表示

```
print(generated_text)
```

6. ※実行結果省略:APIキー入力がないと観覧できないため※

```
In [ ]: #プログラムテキスト生成(無料版)
#promptを用意する

prompt="リスト内の数字を合計するPythonプログラムを教えてください。"
```

```
In [ ]: import openai

# GPT-3モデルを指定してテキスト生成をする
response = openai.Completion.create(
    model="davinci",
    prompt=prompt,
    temperature=0.7
```

```

)
# 生成されたテキストを表示する
print(response['choices'][0]['text'])

```

解答例

```
import itertools
```

```
In [ ]: #API有料版のプログラムテキスト生成の例
import openai

# OpenAI APIを使用するためにAPIキーを設定する
api_key = "Set your OpenAI API key" # ここに実際のAPIキーを入力してください
openai.api_key = api_key
```

```
# GPT-3エンジンを指定してプログラム生成のAPIリクエストを作成する
response = openai.Completion.create(
    engine="davinci",
    prompt="リスト内の数字を合計するPythonプログラムを教えてください。",
    max_tokens=100,
    temperature=0.7,
    api_key=api_key,
)
```

```
# APIリクエストの結果から生成されたプログラムを取得する
generated_code = response['choices'][0]['text']

# 生成されたプログラムをコンソールに表示する
print(generated_code)
```

#表示結果の例↓

```
#import math g = [ 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 ]
#解答例
#$ python3
```

```
import math g = [ 13 , 14 , 15 , 16 , 17 , 18 , 19 , 20 , 21 , 22 , 23 ] d
ef sum (g): L = 0 count = 0 for i in g: if i == 0 : L += 1 else : L += i c
ount += 1 return L + (count * math . sqrt(count) * 1.5 ) print (sum(g))
```

解答例

```
$ python3
```

チャート・テキスト生成

チャートの場合、「prompt」は「messages」に変えて指定する

role: system, content という内容の指定をする必要がある

role: user, content というユーザー入力内容を指定する必要である

```
messages = [
    {"role": "system", "content": "茜は女子高生な妹キャラのチャッ
トAIです。おにいちゃんと会話します。"},

    {"role": "user", "content": "おはよう"},
```

]

モデルエンジンは「gpt-3.5-turbo」に変わる

チャートの例：

- system:チャートAIの振る舞いに対する指示

「system」 :女子学生キャラクターのチャートAIです。

- user:人間の発話

「user」 :おはよう

- 出力:AIの発話

「出力」 :おはようございます。おにいちゃん。今日はどんな予定がありますか？

openai.ChatCompletion.create()のパラメーター説明

チャートパラメーターの解説:

1. 編集パラメーターを指定

```
response = openai.ChatCompletion.create(
```

2. 使用するGPT-3エンジンを指定

```
model="gpt-3.5-turbo",
```

3. チャートのパラメーターを定義

```
messages=messages,
```

4. 出力の多様性を制御するための温度パラメーターを指定

※デフォルト: 1、0~2範囲内に設定※

```
temperature=0)
```

```
In [ ]: # チャットメッセージリストの準備
messages = [
    {"role": "system", "content": "茜は女子高生な妹キャラのチャットAIです。おにいちゃんがおしゃべりをしてくれます。"},
    {"role": "user", "content": "おはよう"},
```

```
In [ ]: import openai

# チャットの実行
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
```

```
    messages=messages,
    temperature=0
)
response["choices"][0]["message"]["content"]

#表示結果の例↓
# おはようございます!おにいちゃん、今日は何をする予定ですか?
```

Out[]: 'おはようございます!おにいちゃん、今日は何をする予定ですか?'

テキスト挿入の生成

モデルエンジンは「text-davinci-003」または「davinci」に変わる

テキスト挿入の場合、「prompt」は「prefix_prompt」と「suffix_prompt」に変えて指定する

original_text:元の文章

元のテキスト

```
original_text = "こんにちは、私はOpenAIのAPIを使って文章生成を行っています。"
```

前後のプロンプト（入力文章）を指定

prefix_prompt: 挿入テキスト「以前」のプロンプト（入力文章）

suffix_prompt: 挿入テキスト「以後」のプロンプト（入力文章）

```
prefix_prompt = "以下の文章に、適切な続きを挿入してください。\\n\\n"
suffix_prompt = "\\nこの続きは、元の文章にうまく合うようにしてください。"
```

出力

元のプロンプトに対し prefix_prompt 指定通りに適切な続きをAIが返してくれる、以下の「これは、私が書いたコードです。」はAIの挿入テキストである

*真ん中の終わってない文章「このコードは、文章を生成するために、私た」はトークン数の制限が50だったため

suffix_promptはそのままうしろに表示される

こんにちは、私はOpenAIのAPIを使って文章生成を行っています。これは、私が書いたコードです。

このコードは、文章を生成するために、私た
この続きは、元の文章にうまく合うようにしてください。

テキスト挿入生成のプロンプト指定および実行例をいくつか挙げる:

例1：元のテキストに対し後ろに prefix_prompt で指示した内容を suffix_prompt の前に挿入する例

例2: prefix_prompt でプログラム内容を指示し、結果を生成させ suffix_prompt の前に結果を挿入する例

例3: prefix_prompt でプログラム内容を指示し、そのプログラムを生成させ suffix_prompt の前にコード表示とともに足りない内容を挿入させる例

```
In [ ]: #例1: テキスト挿入プロンプトの準備
prefix_prompt = "以下の文章に、適切な続きを挿入してください。\\n\\n"
suffix_prompt = "\\nこの続きを、元の文章にうまく合うようにしてください。"
original_text = "こんにちは、私はOpenAIのAPIを使って文章生成を行っています。"
```

```
In [ ]: import openai

# 挿入の実行
response = openai.Completion.create(
    engine="davinci",
    prompt=prefix_prompt + original_text + "\\n\\n",
    max_tokens=50,
    temperature=0.6
)

generated_text = response.choices[0].text.strip()
print(original_text + generated_text + suffix_prompt)

#表示結果の例↓
#
generated_text = response.choices[0].text.strip()
print(original_text + generated_text + suffix_prompt)

#表示結果の例↓
# こんにちは、私はOpenAIのAPIを使って文章生成を行っています。これは、私が書いたコードです。

# このコードは、文章を生成するために、私た
# この続きを、元の文章にうまく合うようにしてください。
```

こんにちは、私はOpenAIのAPIを使って文章生成を行っています。これは、私が書いたコードです。

このコードは、文章を生成するために、私た
この続きを、元の文章にうまく合うようにしてください。

```
In [ ]: #例2: プログラム結果挿入プロンプトの準備
prefix_prompt = "以下は、与えられたリスト内の数値を合計するPythonのプログラムです。\\n\\n"
suffix_prompt = "\\n上記のプログラムは、与えられたリスト内の数値を合計する簡単な例です。"
```

```
In [ ]: import openai

openai.api_key = "Set your OpenAI API key"
```

```

#挿入の実行
response = openai.Completion.create(
    model="text-davinci-003",
    prompt=prefix_prompt + "\n\n",
    temperature=0.7,
    max_tokens=50
)
generated_text = response.choices[0].text.strip()
print(generated_text + suffix_prompt)

[11]
)
generated_text = response.choices[0].text.strip()
print(generated_text + suffix_prompt)

#表示結果の例↓
# 実行結果:
# 合計: 15
# 上記のプログラムは、与えられたリスト内の数値を合計する簡単な例です。

```

実行結果:

合計: 15

上記のプログラムは、与えられたリスト内の数値を合計する簡単な例です。

```

In [ ]: #例3: プログラムや結果挿入の例
import openai

openai.api_key = "Set your OpenAI API key"

prefix_prompt = "以下は、与えられたリスト内の数値を合計するPythonのプログラムです。\\n\\n"
suffix_prompt = "\\n上記のプログラムは、与えられたリスト内の数値を合計する簡単な例です。"

response = openai.Completion.create(
    engine="davinci",
    prompt=prefix_prompt,
    max_tokens=150,
    stop=None
)

generated_code = response.choices[0].text
print(prefix_prompt + generated_code + suffix_prompt)

#表示結果の例↓

# 以下は、与えられたリスト内の数値を合計するPythonのプログラムです。

# ``python
# def sum_list_numbers(numbers):
#     total = 0
#     for num in numbers:
#         total += num
#     return total

# input_numbers = [1, 2, 3, 4, 5]
# result = sum_list_numbers(input_numbers)
# print('合計:', result)
# ``

# フォーマットに対応した字句解析を行うために、次のコードはPythonのf-stringプログラマを使っています

```

```
# `python`引数: ${(引用符合正式表現でこのコードの`print()`を引数にすると、囲まれた式を値
# 上記のプログラムは、与えられたリスト内の数値を合計する簡単な例です。}
```

以下は、与えられたリスト内の数値を合計するPythonのプログラムです。

```
```python
def sum_list_numbers(numbers):
 total = 0
 for num in numbers:
 total += num
 return total

input_numbers = [1, 2, 3, 4, 5]
result = sum_list_numbers(input_numbers)
print('合計:', result)
```

```

フォーマットに対応した字句解析を行うために、次のコードはPythonのf-stringプログラマを使っています。

```
`python`引数: ${(引用符合正式表現でこのコードの`print()`を引数にすると、囲まれた式を値に
設定して Pythonの`print()`呼び出しで変数\$

上記のプログラムは、与えられたリスト内の数値を合計する簡単な例です。
```

編集テキスト生成

モデルエンジンは「text-davinci-edit-001」に変わる

テキスト編集の場合、「prompt」は「input」と「instruction」に変えて指定する

前後のプロンプト（入力文章）を指定

input: 入力テキスト（編集したいテキスト文）

instruction:編集指示

例：

```
input = "今日はとても楽かった。"
instruction = "誤字脱字を修正してください。"
```

openai.Edit.create()の作成パラメーターの説明

パラメーターの解説:

1. 編集パラメーターを指定

```
openai.Edit.create(
```

2. 使用するGPT-3エンジンを指定

```
model="text-davinci-edit-001",
```

3. 入力や編集パラメーターを定義

```
    input=input,  
    instruction=instruction,
```

4. 出力の多様性を制御するための温度パラメーターを指定

※デフォルト: 1、0~2範囲内に設定※

```
    temperature=0)
```

```
In [ ]: # テキスト編集プロンプトの準備  
input = "今日はとても楽かった."  
instruction = "誤字脱字を修正してください。"
```

```
In [ ]: import openai  
  
# 編集の実行  
response = openai.Edit.create(  
    model="text-davinci-edit-001",  
    input=input,  
    instruction=instruction,  
    temperature=0  
)  
print(response["choices"][0]["text"])  
  
#表示結果の例↓  
# 今日はとても楽しかった。
```

今日はとても楽しかった。

画像生成について

公式サイト

<https://platform.openai.com/docs/guides/images>

画像生成には3つの機能がある:

- テキストからの画像生成
- 画像とテキストからの画像編集
- 画像からのバリエーション生成

利用料金:

OpenAI Playgroundの利用料金と同じである

常に実行するコードセットを用意し、利用パラメーターも料金プラン※範囲内に入るよう設定しておく必要がある

設定パラメーター料金プランはOpenAI Playgroundの利用料金・トークン数についてをご参照ください

尚) 各自「生成に要するパラメーター」は決まっているので各自のパラメーター基準表に従ってください

`openai.Image.create()`: 画像生成パラメーター

`openai.Image.create_edit()`: 画像編集パラメーター

`openai.Image.create_variation()`: バリエーション画像パラメーター

`openai.Embedding.create()`: 埋め込み画像パラメーター

尚) 各自生成出力に要する「レスポンスパラメーター」は決まっているので各自のパラメーター基準表に従ってください

`openai.Image.create()`: 画像生成のレスポンスパラメーター

`openai.Image.create_edit()`: 画像編集のレスポンスパラメーター

`openai.Image.create_variation()`: バリエーション画像のレスポンスパラメーター

`openai.Embedding.create()`: 埋め込み画像のレスポンスパラメーター

`openai.Image.create()` のパラメーター

| パラメーター | 説明 |
|------------------------------|--|
| <code>prompt</code> | 画像内容テキストは1000文字まで |
| <code>n</code> | 生成画像の枚数は10枚まで、デフォルト:1 |
| <code>size</code> | 生成画像のサイズ256x256/512x512/1024x1024/ デフォルト:1024x1024 |
| <code>response_format</code> | 生成画像の形式(url/b64_json) |
| <code>user</code> | エンドユーザーID |

`openai.Image.create_edit()` のパラメーター

| パラメーター | 説明 |
|---------------------|--|
| <code>image</code> | 編集する画像は4MB未満の正方形写真 (PNG(RGBA)) |
| <code>mask</code> | 編集エリアのマスク画像は4MB未満の正方形写真 (PNG(RGBA)) |
| | 編集領域に透明色を指定 |
| <code>prompt</code> | 画像内容テキストは1000文字まで |
| <code>n</code> | 生成画像の枚数は10枚まで、デフォルト:1 |
| <code>size</code> | 生成画像のサイズ256x256/512x512/1024x1024/ デフォルト:1024x1024 |

| パラメーター | 説明 |
|-----------------|-----------------------|
| | ト:1024x1024 |
| response_format | 生成画像の形式(url/b64_json) |
| user | エンドユーザーID |

前準備:

1. Colab ノートブックを開く
2. openaiパッケージをインストールする

```
!pip install openai
```

3. 環境変数の準備をする(有料版、無料版の場合は不要)

```
import os
os.environ["OPENAI_API_KEY"] = "<OpenAI_APIのAPIキー>"
```

入力（プロンプト）の準備:

4. 予めに「prompt = 画像内容を入力」を以下のように書いて準備しておく必要がある

```
prompt = "A colorful trees with fruits over the mountains."
```

画像生成に要するパラメーター例:

5. テキストからの画像生成の実行する画像内容はprompt通りと指定

6. 生成枚数やサイズを指定

```
import openai

response = openai.Image.create(
    prompt=prompt,
    n=1,
    size="512x512"
)
```

7. 画像のurl出力を指定

```
image_url = response["data"][0]["url"]
```

8. 画像urlの表示命令

```
print(image_url)
```

```
print(response)
```

9. 画像・urlが出力される

クリックすると生成画像が表示される

<https://oaidalleapiprodsus.blob.core.windows.net/private/org-NjF1gp1qnlpNzEmqp6EJAW3q/user-c3zZjM7KFAtCNeD4qSCzR7mv/img-PMq7flnnKFR8s90X1lU8hFUB.png?st=2023-08-09T00%3A09%3A04Z&se=2023-08-09T02%3A09%3A04Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rsct=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&ske=2023-08-08T18%3A42%3A13Z&sks=b&skv=2021-08-09T18%3A42%3A13Z&sig=WcQV0cNTFplsXAPadvLRxvMsXvaUpQMo6Tx04sAhLtU%3D>

```
In [ ]: #例1：画像生成プロンプトの準備
import openai

openai.api_key = "your_API_key"

# Example prompt
prompt = "A colorful trees with fruits over the mountains."

# Generate the image
response = openai.Image.create(
    prompt=prompt,
    n=1,
    size="512x512"
)
image_url = response["data"][0]["url"]

# Open the image in a web browser
print(image_url)
```

<https://oaidalleapiprodsus.blob.core.windows.net/private/org-NjF1gp1qnlpNzEmqp6EJAW3q/user-c3zZjM7KFAtCNeD4qSCzR7mv/img-NFo5D97C8cwYubeTbsmil1Eo.png?st=2023-08-28T02%3A54%3A17Z&se=2023-08-28T04%3A54%3A17Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rsct=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&ske=2023-08-28T17%3A06%3A57Z&sks=b&skv=2021-08-06&sig=FFxCvYUT8BZsA4AWGJdlitl0k4HvwoSbpLE5NIwCK7E%3D>

```
In [ ]: from IPython.display import Image

# Display the image using its URL
Image(url=image_url)

#図4.5 出所: OpenAI生成画像 |A colorful trees with fruits over the mountains
```

```
In [ ]: #例2：画像生成プロンプトの準備(無料版)
prompt = "cat dancing on car"
```

```
In [ ]: import openai

# テキストからの画像生成の実行
response = openai.Image.create(
    prompt=prompt,
    n=1,
```

```
    size="512x512"
)
image_url = response["data"][0]["url"]
print(image_url)
```

```
https://oaidalleapiproscus.blob.core.windows.net/private/org-NjF1gp1qnlpN
zEmqp6EJAW3q/user-c3zZjM7KFAtCNeD4qSCzR7mv/img-wQMHLbwhs5zyuh8dHYv3R2aK.bn
g?st=2023-08-28T02%3A55%3A44Z&se=2023-08-28T04%3A55%3A44Z&sp=r&sv=2021-08-
06&sr=b&rscd=inline&rsct=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b
067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&skt=2023-08-27T16%3A49%3A57
Z&ske=2023-08-28T16%3A49%3A57Z&sks=b&skv=2021-08-06&sig=rVMp5VzVqMsMpIJw/A
WI4WIc4ZBWl6xF7lgFw%2BDhG80%3D
```

```
In [ ]: from IPython.display import Image

# Display the image using its URL
Image(url=image_url)

#図4.6 出所: OpenAI生成画像 |cat dancing on car
```

```
Out[ ]: No description has been provided for this image
```

画像とテキストからの画像編集

1. 入力（プロンプト）の準備:

予めに「prompt = 画像内容を入力」を以下のように書いて準備しておく必要がある

```
prompt = "A colorful trees with fruits over the
mountains."
```

2. 編集写真の指示

対象の画像の定義「image = ".png", "rb"」と編集エリアをマスクして定義「mask = ".png", "rb"」をする

```
image = open("image.png", "rb")
mask = open("mask.png", "rb")
```

3. 画像ファイルの準備について

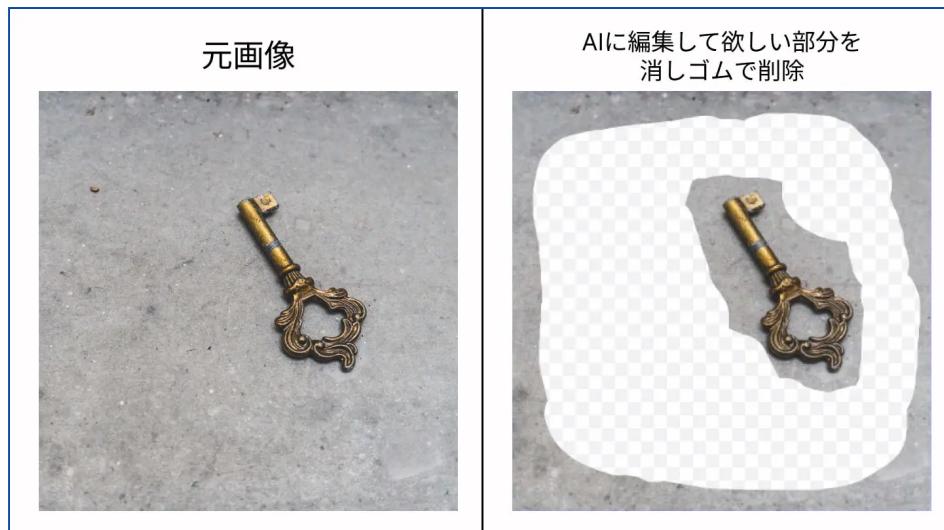


図4.7 出所: OpenAI GPT-4 ChatGPT LangChain 人口知能プログラミング実践入門 |画像編集用写真 |画像編集用マスク写真

編集用写真の準備について :

お好みの画像を使って編集写真とマスク写真を作成

「第2章のインペインティング機能ご参照」

またはインペインティングせずにダウンロードすることも可能

<https://www.borndigital.co.jp/book/>

ダウンロードファイル名

image.png

mask.png

または本画面にでている写真からコピーが可能

3-1. 写真(図4.7)(図4.8)を右クリック

3-2. 名前つけて画面を保存(image.png)

3-3. この写真を各自のPCに保存

3-4. Colab ファイルの左のフォルダを押す

3-5. ↑マークを押す

3-6. 編集写真 image.pngファイルをアップロード

3-7. 以下のコードでダウンロード

```
In [ ]: #ダウンロード前にインストールする
!pip install Pillow
!pip install --upgrade Pillow

Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-pa
ckages (9.4.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-pa
ckages (9.4.0)
Collecting Pillow
  Downloading Pillow-10.0.0-cp310-cp310-manylinux_2_28_x86_64.whl (3.4 MB)
  ━━━━━━━━━━━━━━━━ 3.4/3.4 MB 36.3 MB/s eta 0:0
0:00
Installing collected packages: Pillow
  Attempting uninstall: Pillow
    Found existing installation: Pillow 9.4.0
    Uninstalling Pillow-9.4.0:
      Successfully uninstalled Pillow-9.4.0
Successfully installed Pillow-10.0.0
```

```
In [ ]: #図4.7 元の画像をアップロード
#Colab ファイルの左のフォルダを押す->↑マークを押す->編集.pngファイルをアップロード->以
from PIL import Image
from IPython.display import display

# 画像ファイルのパス
image_path = "/content/image.png" # 画像ファイルの実際のパスを指定してください

# 画像を開く
image = Image.open(image_path)

# 画像の表示
display(image)
```



```
In [ ]: # 画像の情報を表示
print("Image format:", image.format)    # 画像フォーマット(JPEG, PNGなど)
print("Image size:", image.size)         # 画像サイズ(幅 × 高さ)
print("Image mode:", image.mode)        # 画像モード(RGB, Lなど)
```

```
Image format: PNG
Image size: (512, 512)
Image mode: RGB
```

```
In [ ]: #図4.8 編集マスク画像をアップロード
#Colab ファイルの左のフォルダを押す->↑マークを押す->編集.pngファイルをアップロード->以降
from PIL import Image
from IPython.display import display

# 画像ファイルのパス
image_path = "/content/mask.png"  # 画像ファイルの実際のパスを指定してください

# 画像を開く
mask = Image.open(image_path)

# 画像の表示
display(mask)
```



画像生成に要するパラメーター例:

4. テキストからの画像生成の実行する画像内容はprompt通りと指定

生成枚数やサイズを指定

```
import openai

response = openai.Image.create(
    prompt=prompt,
```

5. 編集用コーディングも追加、他のコーディングは変わらない

```
    image=image,
    mask=mask,
    n=1,
    size="512x512"
)
```

6. 画像のurl出力を指定

```
image_url = response["data"][0]["url"]
```

7. 画像urlの表示命令

```
    print(image_url)
```

```
    print(response)
```

8. 画像・urlが出力される

クリックすると生成画像が表示される

```
In [ ]: # 画像とマスクの準備（画像とマスクをカレントフォルダに置いておきます）
image = open("image.png", "rb")
mask = open("mask.png", "rb")
```

```
In [ ]: import openai

# 画像とテキストからの画像編集の実行
response = openai.Image.create_edit(
    image=image,
    mask=mask,
    prompt="a few beautiful apples in cardboard box",
    n=1,
    size="512x512"
)
image_url = response["data"][0]["url"]
print(image_url)
```

```
https://oaidalleapiprodskus.blob.core.windows.net/private/org-NjF1gp1qnlpNzEmqp6EJAW3q/user-c3zZjM7KFAtCNeD4qSCzR7mv/img-BTXw0Ma6f5VlBkcgxbMgmX8h.png?st=2023-08-28T02%3A50%3A06Z&se=2023-08-28T04%3A50%3A06Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rsct=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&skt=2023-08-27T16%3A28%3A20Z&ske=2023-08-28T16%3A28%3A20Z&sks=b&skv=2021-08-06&sig=GJyrMpsS/c/D29abpy940HWYQlAEHNEYrJ0aymwCic%3D
```

```
In [ ]: from IPython.display import Image

# Display the image using its URL
Image(url=image_url)

#図4.9 出所: OpenAI GPT-4 ChatGPT LangChain 人口知能プログラミング実践入門 | 編集後
```

```
Out[ ]:  No description has been provided for this image
```

画像からのバリエーション生成

画像生成と同様に設定するが、生成の指定パラメーターはバリエーション機能の指示にするため「`response = openai.Image.create_variation()`」に変わっただけで他はやることが一緒なので「画像生成の例をご参照ください」

openai.Image.create_variation()のパラメーター

| パラメーター | 説明 |
|--------|--------------------------------|
| image | 編集する画像は4MB未満の正方形写真 (PNG(RGBA)) |
| n | 生成画像の枚数は10枚まで、デフォルト:1 |

| パラメーター | 説明 |
|-----------------|--|
| size | 生成画像のサイズ256x256/512x512/1024x1024/ デフォルト:1024x1024 |
| response_format | 生成画像の形式(url/b64_json) |
| user | エンドユーザーID |

```
In [ ]: # 画像とマスクの準備
image = open("image.png", "rb")
```

```
In [ ]: import openai

# 画像からのバリエーション生成
response = openai.Image.create_variation(
    image=image,
    n=1,
    size="512x512"
)
image_url = response["data"][0]["url"]
print(image_url)
```

<https://oaidalleapiproscus.blob.core.windows.net/private/org-NjF1gp1qnlpNzEmqp6EJAW3q/user-c3zZjM7KFAtCNeD4qSCzR7mv/img-mn80qV83TrkzYeKmrmp1S1Yl.png?st=2023-08-28T02%3A50%3A25Z&se=2023-08-28T04%3A50%3A25Z&sp=r&sv=2021-08-06&sr=b&rscd=inline&rsct=image/png&skoid=6aaadede-4fb3-4698-a8f6-684d7786b067&sktid=a48cca56-e6da-484e-a814-9c849652bcb3&skt=2023-08-27T17%3A05%3A28Z&ske=2023-08-28T17%3A05%3A28Z&sks=b&skv=2021-08-06&sig=/S6EaTJ5E9kpyduC11w5vy3PbLAISgemQx05z9Jihr0%3D>

```
In [ ]: from IPython.display import Image

# Display the image using its URL
Image(url=image_url)

# 図4.10 出所: OpenAI GPT-4 ChatGPT LangChain 人口知能プログラミング実践入門 | バリ
```

Out[]: No description has been provided for this image

埋め込みデータの生成について

公式サイト

Embeddings-OpenAI API

<https://platform.openai.com/docs/guides/embeddings/>

クラスタリングやレコメンドなどのデータ分析に活用する埋め込み (Embedding) 生成:

テキストをコンピュータが処理しやすい形式で表現するための手法である

テキストを「ベクトル表現」（浮動小数配列）に変換する

ベクトル表現の目的:

- 検索：クリエイティブ文字列との関連性によって結果をランクづけ
- クラスタリング：テキストの関連性によってグループ化
- レコメンド：テキストの関連性によってコメント
- 異常検出：類似度がほとんどない外れ値を検出
- 多様性測定：類似度分布を分析
- 分類：テキストが最も類似したラベルによって分類

利用料金:

「テキスト生成」「画像生成」と同様に「埋め込み生成」も「OpenAI API」の利用料金に参考できる

Embeddingモデル:

text-embedding-ada-002

openai.Embedding.create()のパラメーター

| パラメーター | 説明 |
|--------|----------------------|
| model | モデルID |
| input | 埋め込み生成する文字列、または文字列配列 |
| user | エンドユーザーID |

前準備:

テキスト生成と同じ前準備をする

1. パッケージのインストール

```
!pip install openai
```

2. 環境変数の準備

```
import os
os.environ["OPENAI_API_KEY"] = "Set your OpenAI API key"
```

3. 埋め込みに変換するテキストの準備

```
text = "これはテストです。"
```

埋め込み生成の例:

テキストを埋め込みに変換する

4. パラメーターを定義

```
response = openai.Embedding.create(
```

5.変換元の定義

```
    input=text,
```

6.モデルを定義

```
        model="text-embedding-ada-002"  
    )
```

7.表示命令

```
print(len(response["data"][0]["embedding"]))  
print(response["data"][0]["embedding"])
```

8.表示結果

「テスト」という文字をコンピュータが処理しやすい形式で「ベクトル表現」または(浮動小数配列)に変換されて以下のように結果が表示される

```
[-0.0018796126823872328, -0.011050041764974594, -0.00236902735196054,  
-0.01619458757340908, -0.007733077742159367, 0.005056743510067463,...
```

```
In [ ]: # パッケージのインストール  
!pip install openai
```

```
In [ ]: # 環境変数の準備  
import os  
os.environ["OPENAI_API_KEY"] = "Set your OpenAI API key"
```

```
In [ ]: # 埋め込みに変換するテキストの準備  
text = "これはテストです。"
```

```
In [ ]: import openai  
  
# テキストを埋め込みに変換  
response = openai.Embedding.create(  
    input=text,  
    model="text-embedding-ada-002"  
)  
  
# 確認  
print(len(response["data"][0]["embedding"]))  
print(response["data"][0]["embedding"])  
  
# 表示結果の例↓  
# 1536次元の「ベクトル表現」結果が得られる  
# 1536  
# [-0.0018796126823872328, -0.011050041764974594, -0.00236902735196054, -
```

$[-0.0018796126823872328, -0.011050041764974594, -0.00236902735196054, -0.01619458757340908, -0.00773077742159367, 0.005056743510067463, -0.021059466525912285, -0.01351500116288662, 0.00506324740126729, -0.024662600830197334, 0.003856783267110586, 0.015179986134171486, -0.0017869328148663044, -0.030776219442486763, -0.0024405696894973516, -0.009885852225124836, 0.02275046892464161, -0.012415850535035133, 0.01339793112128973, -0.0378263927961586, 0.018614020198583603, -0.0008373706950806081, -0.01237682718783617, 0.000515430117957294, -0.037462178617715836, 0.0011861396487802267, 0.01802867278456688, -0.03415822237730026, 0.0040323869325220585, -0.017950626090168953, 0.024376431480050087, -0.007785108871757984, -0.015401117503643036, -0.021683836355805397, 0.005668105557560921, -0.023752061650156975, 0.007843643426895142, -0.002243828261271119, -0.00019145708938594908, 0.0067900195717811584, 0.02477966994047165, -0.005677861161530018, 0.012311788275837898, -0.004299045074731115, 0.004975445568561554, -0.0038372715935111046, 0.00255763903260231, -0.005895740352571011, -0.0023235005792230368, 0.01732625626027584, 0.013580039143562317, -0.0279405415058136, -0.04451235011219978, -0.015921425074338913, 0.004477900918573141, -0.0038047523703426123, -0.006311986595392227, 0.015609240159392357, 0.0028324266895651817, -0.02229519933462143, -0.009638706222176552, 0.008916778489947319, -0.014711708761751652, 0.004338067956268787, 0.007934696972370148, -0.004451885353773832, -0.019004249945282936, 0.005827449727803469, -0.020083889365196228, 0.010698833502829075, 0.0017706732032820582, 0.022009029984474182, 0.008487524464726448, -0.012558935210108757, 0.04482453688979149, 0.006972127594053745, -0.02657473273575306, -0.012519911862909794, 0.015713302418589592, 0.001256869058124721, 0.016103534027934074, -0.007420893292874098, -0.017248211428523064, -0.01780754141509533, -0.004594970028847456, -0.007134723477065563, -0.020305020734667778, 0.018614020198583603, 0.004416114185005426, -0.01228577271103859, 0.009619194082915783, 0.00227797357365489, 0.004338067956268787, 0.007824132218956947, 0.0017088867025449872, 0.003160871099680662, 0.006429056171327829, 0.038190606981515884, -0.01910831220448017, -0.04326361045241356, -0.01714414916932583, 0.002175537869334221, 0.00866312813013792, 0.012246750295162201, -0.039075132459402084, 0.002969007706269622, 0.014061324298381805, -0.006646934896707535, 0.012916646897792816, -0.016597826033830643, -0.013983277603983879, 0.0030047788750380278, -0.014087339863181114, -0.042196981608867645, -0.028773033991456032, 0.01498487126082182, 0.022217152640223503, -0.034782592207193375, -0.025273961946368217, -0.004133197013288736, 0.013475977815687656, 0.004406358581036329, 0.02182692103087902, 0.0002534469240345061, 0.007186754606664181, 0.009788294322788715, -0.023478899151086807, -0.02277648262679577, -0.006243696436285973, -0.01782055012881756, 0.01895221881568432, 0.0008601341978646815, 0.01172644179314375, 0.011934565380215645, -0.01826281100511551, 0.009385055862367153, -0.012487392872571945, 0.020409081131219864, -0.02861694060266018, -0.0017755511216819286, 0.024025222286581993, 0.001912132021971047, -0.014659678563475609, -0.030464034527540207, -0.010211044922471046, 0.008838732726871967, 0.013931247405707836, 0.004533183760941029, 0.009033847600221634, -0.008357447572052479, -0.00534291286021471, -0.01025657169520855, 0.02614547871053219, -0.004445381462574005, 0.0031364818569272757, 0.02428537607192993, 0.010334618389606476, 0.00714122736826539, -0.0157393179833889, -0.017287233844399452, 0.001990178134292364, 0.021670829504728317, 0.026847893372178078, -0.019290419295430183, 0.013098753988742828, 0.03595328330993652, 0.004471397027373314, 0.018353864550590515, 0.008598090149462223, 0.0002195049455622211, -0.029345372691750526, 0.01917335018515587, -0.015648264437913895, 0.018171757459640503, -0.017872581258416176, 0.030584356281906366, -4.743237514048815e-05, -0.003482811851426959, 0.0029218546114861965, -0.010750864632427692, -0.0386849008500576, 0.007303823716938496, 0.006920096464455128, -0.005144545342773199, 0.0032096500508487225, -0.011206134222447872, 0.011954076588153839, 0.0033950097858905792, 0.005833140457980335, -0.012767057865858078, 0.03290948271751404, 0.037930455058813095, -0.0051575531251728535, 0.004435625858604908, -0.6431007385253$

906, -0.007329839281737804, 0.01013299822807312, -0.006146138533949852, 0.027186093851923943, 0.0344183743000305, 0.029631542041897774, 0.012025618925690651, -0.029215294867753983, 0.03134855628013611, -0.002320248633623123, -0.004799841437488794, -0.01753438077867031, -0.016597826033830643, -0.013196311891078949, -0.02676984667778015, -0.00902083981782198, -0.00819485168904066, 0.010718345642089844, 0.021930983290076256, -0.011440272442996502, 0.025976378470659256, -0.004754314664751291, -0.008058270439505577, 0.007446908392012119, 0.013306877575814724, 0.012175207957625389, -0.014334485866129398, 0.005876228678971529, 0.003482811851426959, -0.005957526620477438, 0.007655031513422728, -0.0025088603142648935, 0.004312052857130766, 0.034548453986644745, -0.004497412592172623, -0.014711708761751652, 0.050795068964362144, 0.0002418619260424748, 0.026639770716428757, -0.020200958475470543, -0.01618157885968685, 0.019186358898878098, -0.0008288344251923263, 0.009502124972641468, 0.014165385626256466, 0.007739581633359194, -0.03519883751869202, 0.016103534027934074, 0.03421025350689888, -0.023036638274788857, -0.01012649480253458, 0.002251958241686225, -0.003362490562722087, 0.006344506051391363, -0.007908682338893414, 0.017235202714800835, -0.006159146316349506, -0.004416114185005426, 0.0005231534596532583, -0.03449642285704613, 0.012565438635647297, 0.008357447572052479, -0.0034567962866276503, -0.0238951463252306, 0.0169100109487772, -0.031166451051831245, 0.02297159843146801, 0.015166978351771832, -0.0038015004247426987, -0.016571810469031334, -0.011602869257330894, 0.010965491645038128, -0.0066339271430788, 0.03803451731801033, 0.022113090381026268, 0.023283783346414566, -0.009879348799586296, -0.015726309269666672, 0.009027344174683094, -0.002396687731444836, -0.0069916388019919395, -0.010783383622765541, -0.008136316202580929, 0.003941333387047052, -0.03988160938024521, -0.04165065661072731, -0.012272764928638935, 0.0042209988459944725, 0.0047315508127212524, 0.010789887979626656, 0.008272897452116013, 0.005811190232634544, -0.013176800683140755, 0.0013300373684614897, 0.019875766709446907, -0.00822737067937851, -0.009254978969693184, 0.015127955935895443, -0.0007028222898952663, -0.02476666122674942, -0.00017743316129781306, 0.03176480531692505, 0.04292541183531284, 0.013306877575814724, -0.015232017263770103, -0.013040219433605671, 0.024883730337023735, 0.024025222286581993, -0.015075924806296825, 0.01183050312101841, -0.015023893676698208, -0.015401117503643036, 0.001232479582540691, -0.007655031513422728, -0.041286442428827286, 0.0034730560146272182, 0.020838337019085884, 0.010516725480556488, 0.007193258497864008, 0.035927269607782364, 0.0030519317369908094, 0.05432015657424927, 0.015179986134171486, 0.005281126592308283, 0.018783118575811386, 0.0006707095308229327, -0.0030600614845752716, -0.014893816784024239, 0.004308800678700209, -0.018509957939386368, 0.005599814932793379, 0.012571942061185837, -0.020187951624393463, -0.0026584488805383444, -0.00032173734507523477, 0.0009292375762015581, -0.012266261503100395, 0.02026599645614624, -0.00833051237463951, -0.011537830345332623, -0.0013601176906377077, -0.004422618076205254, 0.0028698239475488663, -0.011980092152953148, -0.004022631328552961, -0.014347493648529053, -0.0020828580018132925, -0.009781790897250175, 0.013118266128003597, 0.00956716388463974, 0.008181843906641006, -0.019069289788603783, 0.008142820559442043, -0.017742503434419632, -0.0025186159182339907, -0.007934696972370148, -0.030490050092339516, -0.009645209647715092, -0.0007190819596871734, 0.011219142004847527, 0.022841522470116615, -0.014035308733582497, 0.0025446314830332994, -0.024493500590324402, -0.01446456275803368, -0.02500079944729805, 0.015245025046169758, -0.02315370738506317, -0.04339369013905525, -0.021501729264855385, -0.003928325604647398, 0.004767322447150946, 0.010269579477608204, 0.007160739041864872, 0.0026909681037068367, -0.005554288160055876, 0.0006804653094150126, -0.011375234462320805, -0.028278740122914314, -0.002224316820502281, 0.003082825103774667, 0.011876030825078487, 0.0157393179833889, 0.004370587412267923, 0.02885107882320881, 0.017937619239091873, 0.025117870420217514, -0.012194719165563583, 0.006770507898181677, -0.014659678563475609, 0.019511550664901733, -0.018145741894841194, 0.018522964790463448, 0.010393152944743633, -0.004565702751278877, 0.0017446578713133931, 0.0047315508127212524, -0.000

13932466390542686, 0.02409026212990284, 0.040375903248786926, 0.0104777030646801, 0.006217680871486664, -0.01732625626027584, 0.019277412444353104, -0.024038231000304222, -0.023101676255464554, 0.007336343172937632, 0.03233714401721954, 0.013736131601035595, -0.007459916174411774, -0.033455803990364075, -0.02570321597158909, 0.009866341017186642, 0.021956998854875565, 0.019472528249025345, -0.02697797119617462, 0.004988453350961208, 0.00023921973479446024, -0.0002506014716345817, 0.0028714498039335012, 0.0010902079520747066, 0.017742503434419632, -0.03106238879263401, -0.019719673320651054, -0.004185227677226067, 0.01713114231824875, 0.026743832975625992, 0.020331036299467087, -0.013053227216005325, -0.019745688885450363, -0.016779933124780655, -0.018106719478964806, 0.017027080059051514, 0.012396338395774364, -0.032545264810323715, 0.018731089308857918, -0.004386846907436848, 0.04081816226243973, 0.024883730337023735, 0.0059347632341086864, 0.017248211428523064, 0.011550838127732277, -0.007258296944200993, 0.029579510912299156, 0.01645474135875702, 0.051926739513874054, 0.024181315675377846, -0.007830635644495487, 0.032545264810323715, -0.012324796058237553, -0.008038759231567383, 0.000549575372133404, -0.018522964790463448, 0.011102071963250637, -0.030438018962740898, -0.00368443108163774, -0.0040323869325220585, 0.013150785118341446, 0.03717600926756859, 0.018861165270209312, 0.03733209893107414, 0.0007381869945675135, -0.00957366731017828, 0.00727130472660647, -0.012071145698428154, 0.011258164420723915, -0.0040323869325220585, -0.010373640805482864, -0.018132735043764114, -0.031660743057727814, -0.012324796058237553, 0.010633795522153378, 0.0017105125589296222, 0.01888718083500862, 0.005118530243635178, 0.030229896306991577, 0.012780065648257732, 0.04092222452163696, -0.00301290862262249, -0.009742767550051212, -0.021267591044306755, 0.009775286540389061, 0.015258032828569412, -0.0007686737808398902, -0.023075660690665245, -0.003440536791458726, -0.019277412444353104, -0.010998010635375977, 0.021501729264855385, 0.005964030511677265, -0.0006950990064069629, 0.014620655216276646, -0.018588004633784294, 0.007349350955337286, 0.0233878456056118, 0.020070882514119148, 0.007824132218956947, -0.04438227415084839, -0.016701888293027878, 0.016949033364653587, -0.011583357118070126, 0.017521372064948082, -0.008110301569104195, 0.01476373989135027, -0.010536237619817257, -0.006425803992897272, -0.0200858426707983, -0.0029543740674853325, -0.009827317669987679, 0.003596629248932004, 0.0120581379160285, -0.012272764928638935, -0.027576325461268425, -0.010965491645038128, 0.005489249713718891, 0.007583489175885916, 0.014685694128274918, 0.020760290324687958, -0.00602581724524498, -0.022321214899420738, -0.009157421067357063, -0.02113751322031021, 0.000895092380233109, 0.019251396879553795, 0.01228577271103859, 0.010074463672935963, 0.0042665256187319756, -0.02069525234401226, -0.007635520305484533, -0.03129652887582779, -0.020200958475470543, 2.7870013582287356e-05, -0.007589993067085743, -0.009937883354723454, 0.019641628488898277, 0.011797984130680561, -0.018757104873657227, 0.008337936364114285, 0.01713114231824875, -0.010464695282280445, -0.0035478502977639437, 0.017885588109493256, 0.0036714235320687294, -0.0164027102291584, 0.0003127945528831333, -0.0013023960636928678, -0.006777011789381504, -0.012793073430657387, -0.024922754615545273, 0.0191346051543951, 0.022113090381026268, 0.018731089308857918, -0.016519779339432716, -0.004308800678700209, 0.004351075738668442, -0.0169100109487772, 0.014100347645580769, -0.0007316831615753472, 0.030516065657138824, -0.01486780121922493, 0.04349774867296219, 0.002942992141470313, 0.0010633794590830803, 0.0024828447494655848, 0.00878019817173481, 0.0010975247714668512, 0.005765663459897041, 0.006777011789381504, -0.01664985716342926, 0.007837139070034027, 0.023530930280685425, -0.012806081213057041, -0.01385320071130991, 0.015947440639138222, -0.00579493073746562, -0.05603717267513275, -0.026691801846027374, 0.004107181448489428, 0.026925940066576004, -0.015440140850841999, -0.0017788030672818422, 0.0007410324178636074, -0.0012942661996930838, -0.017300240695476532, -0.0013495489256456494, -0.009645209647715092, -0.016532788053154945, -0.027862494811415672, -0.0396995022892952, 0.005690868943929672, -0.038814976811409, -0.03332572802901268, -0.017950626090168953, 0.013775154948234558, -0.017274226993322372, -0.0289291255176

06735, -0.0413384735584259, 0.027810463681817055, 0.005941267125308514, 0.02362198382616043, -0.008624105714261532, -0.019186358898878098, 0.028018586337566376, -0.014828778803348541, -0.011323203332722187, 1.4227171959646512e-05, -0.02139766700565815, 0.002464959165081382, 0.015284048393368721, 0.018379880115389824, 0.013593046925961971, 0.011889038607478142, 0.003834019647911191, -0.0063054827041924, 0.006159146316349506, -0.0001585313439136371, 0.012311788275837898, -0.00656888667970896, -0.012454872950911522, 0.007785108871757984, 0.015609240159392357, -0.017456334084272385, -0.007655031513422728, 0.008468013256788254, -0.028929125517606735, -0.000212797851418145, 5.6400574976578355e-05, -0.0025218678638339043, -0.0017056347569450736, 0.015882402658462524, -2.9343542337301187e-05, 0.0024552035611122847, -0.003977104555815458, 0.015843378379940987, -0.016051502898335457, -0.0026226777117699385, 0.0018454674864187837, 0.012545927427709103, 0.0052355993539094925, -0.020435096696019173, 0.025521108880639076, -0.005694120656698942, -0.014672686345875263, 0.0025039822794497013, -0.013098753988742828, 0.03493868187069893, 0.011446776799857616, 0.017261218279600143, -0.0047315508127212524, -0.008864748291671276, -0.026626762002706528, -0.023465892300009727, 0.019888773560523987, -0.00394783727824688, 0.008864748291671276, -0.003170626936480403, -0.022009029984474182, -0.030333956703543663, -0.007694054860621691, -0.016818957403302193, 0.03168675675988197, -8.790360152488574e-05, -0.02952747978270054, -0.013931247405707836, 0.011518319137394428, 0.011505311354994774, 8.241597970481962e-05, 0.0003890115476679057, -0.02546907775104046, -0.00924197118729353, 0.01845792680978775, -0.01372312381863594, 0.0036096367985010147, 0.01914733462035656, 0.013001197017729282, -0.01982373557984829, -0.02120255120098591, 0.013358908705413342, -0.024428460747003555, -0.0029104729183018208, -0.006455071270465851, 0.016949033364653587, 0.013788162730634212, 0.024155300110578537, 0.005404699593782425, -0.0146336629986763, 0.0033397271763533354, -0.002242202404886484, 0.006107115186750889, -0.005206332076340914, 0.013684100471436977, 0.002635685261338949, -0.0034925676882267, 0.01847093552350998, 0.0027722660452127457, 0.0050762551836669445, -0.011889038607478142, 0.009307010099291801, 0.025586146861314774, -0.01136222667992115, 0.012168703600764275, -0.009411071427166462, -0.033221665769815445, 0.013202816247940063, 0.006998142693191767, 0.00552176870405674, 0.0037592253647744656, 0.005430715158581734, -0.018067695200443268, 0.0026308074593544006, -0.0004162464174441993, 0.03059411235153675, 0.013814178295433521, 0.01932944357395172, -0.03223308175802231, -0.006331498268991709, 0.0007784295594319701, -0.0012357315281406045, -0.014464562758803368, -0.0029559999238699675, -0.001448732684366405, -0.009424079209566116, 0.02905920334160328, -0.007238785270601511, 0.010985002852976322, -0.00924197118729353, 0.012272764928638935, 0.010185029357671738, 0.0014747480163350701, -0.007427397184073925, -0.007167242933064699, 0.008487524464726448, 0.010867933742702007, -0.026470670476555824, -0.029111234471201897, -0.015882402658462524, -0.007967216894030571, 0.002051964635029435, 0.03309158980846405, -0.02319272980093956, 0.003411269513890147, -0.009638706222176552, -0.004855124279856682, -0.017053095623850822, 1.143890040111728e-05, 0.02323175221681595, -0.0006780264084227383, 0.038867007941007614, 0.03629148378968239, 0.001104841590858996, -0.024662600830197334, -0.06420601159334183, -0.017495356500148773, -0.006311986595392227, 0.0164677482098341, 0.03709796071052551, -0.004305548965930939, -0.018340857699513435, -0.0039185695350170135, 0.014204408973455429, -0.00789567455649376, -0.010074463672935963, -0.00213814084418118, 0.02115052007138729, -0.01710512675344944, -0.0011820747749879956, 0.018601011484861374, -0.026717817410826683, 0.014477570541203022, -0.0015170230763033032, -0.023374836891889572, -0.021735867485404015, -0.0036811791360378265, -0.006822539027780294, 0.022126099094748497, -0.020773297175765038, 0.008246881887316704, 0.011732946150004864, -0.02141067571938038, -0.001364995609037578, -0.018184764310717583, -0.0283828023821155, -0.006455071270465851, -0.001034925109706819, 0.03231126589775085, 0.006217680871486664, -0.00865662470459938, 0.011771968565881252, 0.007973720319569111, -0.012884126976132393, 0.020305020734667778, -0.02120255120098591, 0.04576108977198601, -

$0.001694252947345376, -0.014412532560527325, -0.018522964790463448, 0.011277676559984684, 0.01797664165496826, -0.009775286540389061, 0.010789887979626656, 0.005138041451573372, -0.013105258345603943, 0.002543005393818021, 0.02654871717095375, -0.014048316515982151, -0.007017654366791248, -0.01409293168783188, -0.01103703398257494, -0.002173912012949586, -0.02094239741563797, -0.014932840131223202, 0.006744492799043655, -0.025299977511167526, -0.01554420217871666, -0.0019983078818768263, -0.013905231840908527, 0.021059466525912285, -0.0062501998618245125, -0.02498779259622097, 0.012721531093120575, 0.025156892836093903, -0.025117870420217514, -0.003518583020195365, -0.011511814780533314, 0.0025169900618493557, 0.012877623550593853, 0.012123176828026772, 0.010698833502829075, -0.015687286853790283, 0.0332476831972599, -0.020864350721240044, 0.0019316434627398849, -0.029371388256549835, -0.008110301569104195, 0.022815506905317307, -0.01273453887552023, 0.009411071427166462, 0.0044909087009727955, 0.004338067956268787, 0.02137165144085884, 0.02046111226081848, -0.011277676559984684, -0.0014690571697428823, -0.002385287079960108, 0.0020031859166920185, 0.012409346178174019, -0.027576325461268425, 0.006181909702718258, 0.011212637647986412, -0.013593046925961971, -0.010536237619817257, 0.0014097095699980855, -0.03179081901907921, 0.0007751776720397174, 0.01495885569602251, -0.018392888829112053, -0.028018586337566376, -0.027810463681817055, 0.000205277708740905, 0.0069461120292544365, -0.01081590261310339, 0.003964096773415804, -0.005459982436150312, -0.004611229989677668, 0.013567031361162663, -0.013124769553542137, 0.01869206503033638, -0.016246618703007698, -0.007629016414284706, 0.0012836974347010255, -0.005957526620477438, -0.024480491876602173, -0.02004486694931984, -0.0045722066424787045, 0.003755973419174552, -0.006998142693191767, -0.025508100166916847, -0.025156892836093903, 0.013697108253836632, -0.04172870144248009, -0.011219142004847527, -0.029189279302954674, 0.012819088995456696, -0.004819353111088276, 0.0027934035751968622, -0.005287630017846823, 0.02768038585782051, 0.017443327233195305, 0.015804355964064598, -0.0382426381111145, -0.014152377843856812, -0.00777210108935833, -0.011511814780533314, -0.004243762232363224, 0.010328114032745361, -0.019693657755851746, -0.024168306961655617, 0.016936026513576508, 0.005203080363571644, 0.01778152585029602, 0.031244495883584023, 0.006829042918980122, -0.022386252880096436, -0.007518450729548931, -0.014139370061457157, -0.00423075444996357, -0.0076875509694218636, -0.003466552123427391, -0.022802498191595078, 0.005007964558899403, 0.008994825184345245, 0.01983674243092537, -0.007648528087884188, 0.019212374463677406, 0.006048580631613731, -0.008975313045084476, 0.014620655216276646, -0.03158269822597504, -0.002635685261338949, -0.016259625554084778, -0.01238333061337471, -0.0015877524856477976, -0.00036868700408376753, 0.012441865168511868, 0.026496686041355133, 0.0028844575863331556, -0.013892224058508873, -0.03355986624956131, 0.023713037371635437, -0.004942926112562418, 0.006542873103171587, 0.012773562222719193, -0.016493763774633408, -0.016818957403302193, 0.0040616546757519245, 0.004373839125037193, -0.015492171049118042, 0.0029234807071475, 0.010965491645038128, -0.013092250563204288, -0.030359972268342972, 0.02479267679154873, 0.039309270679950714, -0.0314786359667778, -0.014919832348823547, -0.004516923800110817, 0.00106907042209059, -0.00524210324510932, -0.022568359971046448, 0.011128087528049946, -0.026184501126408577, 0.03465251252055168, 0.016714895144104958, 0.0017918107332661748, -0.001956033054739237, 0.001274754642508924, -0.0007170494645833969, -0.0209554061293602, -0.0005463234265334904, 0.2761794924736023, -0.01616857200860977, -0.009762278757989407, 0.03106238879263401, -0.005040484014898539, 0.032103005796670914, 0.005895740352571011, -0.0003245827683713287, -0.007830635644495487, 0.02277648262679577, 0.01205163449048996, -0.018275819718837738, 0.007498939521610737, 0.012077650055289268, -0.0029576257802546024, -0.014087339863181114, -0.014724716544151306, -0.007524954620748758, -0.010776880197227001, -0.024675607681274414, 0.033455803990364075, -0.008292408660054207, 0.022880544885993004, 0.005830701906234026, 0.0055510359816253185, -0.013645078055560589, -0.0105882678180933, 0.003320215502753854, 0.010237060487270355, 0.0015755577478557825, -0.018731089308857918, 0.00150808$

02841112018, 0.016519779339432716, -0.010783383622765541, -0.0231797229498
6248, -0.010718345642089844, 0.025586146861314774, 0.017209187150001526,
0.02657473273575306, 0.003320215502753854, 0.014425539411604404, -0.012181
711383163929, 0.013144281692802906, -0.003944585099816322, -0.018939211964
60724, 0.01887417398393154, -0.008962305262684822, -0.009027344174683094,
-0.005320149473845959, 0.013436954468488693, -0.01351500116288662, 0.00585
6717005372047, 0.021462704986333847, 0.0016536039765924215, 0.027368200942
873955, -0.006468079052865505, 0.05059995502233505, -0.028018586337566376,
0.019719673320651054, 0.009625698439776897, 0.006952615920454264, 0.014217
416755855083, -0.04081816226243973, 0.04209291934967041, -0.02428537607192
993, 0.013645078055560589, -0.02861694060266018, 0.008415982127189636, -0.
0020747282542288303, -0.010386648587882519, -0.0059770382940769196, -0.002
495852531865239, 0.02477966994047165, -0.0072973198257386684, -0.026002394
035458565, -0.011095568537712097, 0.03767029941082001, 0.0226984377950429
9, 0.028642956167459488, 0.019875766709446907, -0.005755907390266657, 0.00
4429121967405081, -0.0034567962866276503, -0.007752589415758848, -0.039673
48486185074, -0.026639770716428757, -0.0037819889839738607, -0.02151473611
5932465, 0.0054697380401194096, -0.005889236461371183, -0.0101069835945963
86, 0.000533722246453166, 0.00933952908962965, -0.0338200218975544, -0.00
018810354231391102, 0.023999206721782684, -0.008389966562390327, 0.0176514
49888944626, -0.0018535973504185677, 0.003229161724448204, -0.012318292632
699013, 0.019446512684226036, 0.01981072686612606, 0.0392572395503521, -0.
003388505894690752, 0.0036974388640373945, -0.011453280225396156, -0.00043
37255086284131, 0.000560550601221621, -0.023765068501234055, -0.0099378833
54723454, -0.030724188312888145, 0.0038893024902790785, -0.011440272442996
502, 0.004461641423404217, 0.015153970569372177, -0.008441997691988945, 0.
00891027506440878, -0.001988552277907729, -0.014711708761751652, 0.0235309
30280685425, -0.015466155484318733, -0.025065839290618896, 0.0141653856262
56466, -0.006848554126918316, -0.014113355427980423, -0.01447757054120302
2, -0.022139105945825577, 0.022425275295972824, -0.025976378470659256, 0.
023582961410284042, -0.006679454352706671, 0.034574467688798904, 0.01141425
6878197193, 0.014919832348823547, 0.005720136221498251, 0.0001406457595294
3414, 0.009001328609883785, -0.022451290860772133, 0.011902045458555222,
0.03535493090748787, -0.0008861495880410075, -0.0055835554376244545, 0.009
619194082915783, -0.020148927345871925, -0.015648264437913895, 0.025963369
756937027, -0.018574995920062065, 0.007746085524559021, 0.0016105158720165
491, -0.035693131387233734, -0.021983014419674873, 0.006881073582917452,
0.012331300415098667, 0.03608335927128792, 0.006816035136580467, -0.026899
924501776695, -0.04162464290857315, 0.0070566777139902115, -0.003616140689
700842, -0.03311760351061821, -0.006718477234244347, 0.0400116853415966, -
0.021943990141153336, -0.017950626090168953, 0.00041909184074029326, -0.16
597825288772583, 0.005785174667835236, 0.011440272442996502, -0.0137621471
65834904, 0.006562384776771069, 0.006770507898181677, 0.02684789337217807
8, -0.02431139163672924, 0.0022812255192548037, 0.002209683181717992, 0.02
6951955631375313, -0.02385612204670906, -0.022581368684768677, 0.000409132
830100134, -0.012467880733311176, -0.0027088536880910397, -0.0187831185758
11386, -0.029553495347499847, 0.039075132459402084, 0.017430318519473076,
0.02589833177626133, -0.014893816784024239, 0.008734670467674732, 0.020916
38185083866, 0.021501729264855385, -0.005716884508728981, -0.0137621471658
34904, 0.01982373557984829, 0.010152510367333889, 0.002983641345053911, -
0.019082296639680862, 0.004351075738668442, 0.017222195863723755, 0.017573
4031945467, -0.010191533714532852, -0.011713434010744095, 0.02674383297562
5992, -0.007323335390537977, -0.026847893372178078, 0.004848620388656855,
-0.0021609042305499315, 0.019251396879553795, 0.006461575161665678, -0.006
549376994371414, -0.005716884508728981, 0.015700293704867363, 0.0264836773
27632904, -0.002235698513686657, 0.011420761235058308, -0.0094696059823036
2, 0.011882534250617027, -0.03629148378968239, -0.013892224058508873, -0.0
0390112338215113, 0.02090337499976158, 0.0016129548894241452, -0.02330979
8911213875, 0.033455803990364075, 0.00011503684800118208, 0.01610353402793
4074, 0.007225777488201857, -0.03649960830807686, 0.00100077991373837, 0.0

09963898919522762, -0.010191533714532852, -0.0119475731626153, -0.02116352878510952, 0.023023629561066628, 0.008767190389335155, 0.012584949843585491, 0.003834019647911191, -0.014581631869077682, 0.003388505894690752, -0.01710512675344944, -0.001970666693523526, 0.004022631328552961, -0.013118266128003597, 0.03845076262950897, 0.00979479867964983, 0.002279599430039525, -0.003518583020195365, 0.03233714401721954, 0.0034340329002588987, 0.006629862473346293, 0.016792941838502884, 0.012545927427709103, 0.01974568885450363, 0.015140963718295097, -0.0292673259973526, -0.015609240159392357, 0.013105258345603943, -0.04097425565123558, -0.027186093851923943, -0.02428537607192993, -0.01315728947520256, 0.015570217743515968, -0.016610832884907722, -0.0006690836162306368, -0.006419300101697445, -0.0002804785326588899, -0.01913432776927948, -0.004910407122224569, -0.03197292611002922, 0.02817467972636223, 0.03215503692626953, -0.002120255259796977, -0.00035933771869167686, 0.01958959735929966, 0.031634729355573654, 0.008539555594325066, -0.015375101938843727, -0.02457154542207718, 0.006152642425149679, 0.0218009050466079712, 0.02524794638156891, 0.008123309351503849, 0.00014593014202546328, -0.011342714540660381, 0.010412664152681828, 0.009950891137123108, 0.005599814932793379, -0.008298913016915321, -0.014243432320654392, 0.03431431204080582, -0.0028860834427177906, -0.007746085524559021, -0.04992355406284332, -0.01871808059513569, 0.010165518149733543, 0.05546483397483826, 0.010009425692260265, 0.04055801033973694, 0.010067960247397423, 0.02479267679154873, -0.01090695708990097, 0.01914733462035656, -0.008389966562390327, -0.015219009481370449, -0.027784448117017746, -0.002399920718744397, -0.021072475239634514, 0.003073069266974926, -0.005905495956540108, -0.015817364677786827, -0.020318027585744858, 0.03421025350689888, -0.022841522470116615, -0.02137165144085884, 0.004679520148783922, -0.027108047157526016, -0.016493763774633408, -0.005911999847739935, -0.028538893908262253, 0.0329354964196682, 0.011674411594867706, -0.00184221554081887, 0.011524822562932968, -0.01956358179450035, -0.0021088733337819576, -0.014165385626256466, 0.005007964558899403, 0.0043348162434995174, -0.030438018962740898, -0.0018389635952189565, 0.029787633568048477, 0.0017316500889137387, 0.013124769553542137, 0.011531326919794083, -0.009261482395231724, -0.01692301779985428, -0.009963898919522762, -0.02247730642557144, 0.01557021743515968, 0.018184764310717583, -0.017872581258416176, -0.03665570169687271, -0.02003185823559761, 0.010373640805482864, -0.019940804690122604, -0.013567031361162663, 0.028304755687713623, 0.008058270439505577, 0.018067695200443268, 0.003791744587942958, -0.04591718316078186, 0.002983641345053911, -0.011459783650934696, -0.001476373989135027, -0.00923546776175499, 0.0119475731626153, 0.0037819889839738607, 0.017898594960570335, -0.03215503692626953, -0.007960712537169456, 0.016610832884907722, 0.00264544109813869, -0.005245355423539877, 0.006926600355654955, -0.009736264124512672, 0.028408817946910858, -0.04859676957130432, -0.005512013100087643, -0.01845792680978775, -0.01575232483446598, 0.022568359971046448, -0.0017853068420663476, -0.013144281692802906, -0.008708655834197998, 0.009593179449439049, -0.03449642285704613, 0.009489117190241814, 0.010503717698156834, -0.01113459188491106, 0.00225358409807086, 0.013736131601035595, -0.01776851899921894, -0.0022844774648547173, 0.008383463136851788, 0.022919567301869392, -0.020604196935892105, -0.03405416011810303, 0.013280862011015415, 0.013209319673478603, -0.009807806462049484, 0.019238388165831566, 0.00979479867964983, -0.016506772488355637, -0.02450650744140148, -0.07195860147476196, 0.0052355993539094925, -0.0011893915943801403, -0.009976906701922417, -0.03032420063391328, 0.0022308207117021084, 0.007095700595527887, 0.0016430352116003633, 0.008285905234515667, 0.003352734725922346, -0.025989385321736336, 0.004162464290857315, 0.005407951306551695, -0.017417311668395996, -0.032103005796670914, 0.0003802719875238836, 0.028538893908262253, 0.021059466525912285, 0.017391296103596687, 0.01732625626027584, 0.019095303490757942, -0.009619194082915783, -0.002451951615512371, 0.006698965560644865, -0.015153970569372177, 0.0017414059257134795, -0.0205131433904171, -0.005983542185276747, -0.026522701606154442, -0.010217548348009586, 0.00236902735196054, -0.028486864641308784, -0.007986728101968765, 0.00311046629212796

7, -0.0007178624509833753, 0.0033299713395535946, -0.008357447572052479, 0.0182367954403162, 0.030047787353396416, 0.01819777302443981, -0.006575392559170723, -0.03774834796786308, 0.012493896298110485, -0.007154235150665045, -0.031114419922232628, 0.017716487869620323, -0.030333956703543663, 0.013710116036236286, 0.013410938903689384, 0.02139766700565815, 0.006607912015169859, 0.01181749626994133, -0.013274358585476875, -0.0025007303338497877, 0.005053491797298193, -0.029189279302954674, -0.0004150269378442317, -0.006546125281602144, 0.01260446198284626, -0.009202947840094566, 0.04055801033973694, 0.019316434860229492, 0.009651714004576206, 0.007889170199632645, 0.001304834964685142, 0.007342847064137459, 0.005251858849078417, -0.010087471455335617, 0.01616857200860977, -0.033403776586055756, -0.004845368210226297, 0.012754050083458424, 0.004201487172394991, 0.00820785854011774, 0.01692301779985428, 0.015570217743515968, 0.019277412444353104, -0.014217416755855083, -0.027524294331669807, 0.02838280238211155, 0.00409742584452033, 0.007733077742159367, -0.04071410372853279, 0.02048712782561779, 0.018601011484861374, -0.0020096898078918457, -0.00455919886007905, 0.011349218897521496, -0.006711973343044519, -0.0060095577500760555, -0.017651449888944626, 0.005027476232498884, 0.01250690408051014, 0.01871808059513569, -0.0013097128830850124, 0.019199365749955177, 0.028512880206108093, -0.02541704662144184, 0.03871091827750206, 0.0270560160279274, 0.018627027049660683, -0.00287307589314878, 0.0013975148322060704, -0.0186790581792593, -0.0031901386100798845, -0.00036015070509165525, -0.003762477310374379, -0.0629572719335556, 0.001446293666958809, 0.024701623246073723, 0.012643484398722649, 0.004168968182057142, -0.019290419295430183, 0.007941201329231262, -0.01887417398393154, -0.015557209961116314, -0.025833293795585632, 0.012142688035964966, -0.02520892396569252, 0.02341386117041111, 0.011076057329773903, 0.020877359434962273, 0.02976161800324917, -0.020070882514119148, 0.020617205649614334, -0.011420761235058308, 0.012858112342655659, -0.01731324940919876, 0.007954209111630917, 0.017469340935349464, 0.004959185607731342, 0.014932840131223202, -0.016285641118884087, -0.01779453456401825, -0.006152642425149679, 0.00787616241723299, -0.010054952464997768, 0.004038890823721886, -0.03150464966893196, 0.05921105295419693, 0.01326785422861576, -0.022230159491300583, 0.0054729897528886795, 0.0051543014124035835, 0.0007853399147279561, 0.019043274223804474, 0.015817364677786827, -0.0003367775061633438, -0.015023893676698208, 0.02004486694931984, 0.010991507209837437, 0.004038890823721886, -0.01205163449048996, -0.03649960830807686, 0.012422353960573673, -0.022802498191595078, 0.01916034333407879, 0.009911867789924145, 0.012786570005118847, 0.028044601902365685, 0.007251793053001165, 0.01758641190826893, 0.020565174520015717, -0.0010471198474988341, -0.008383463136851788, 0.029891695827245712, -0.003508827183395624, 0.009580171667039394, 0.0031966425012797117, 0.020799312740564346, 0.019316434860229492, -0.03626547008752823, -0.017222195863723755, 0.010536237619817257, -0.011004514992237091, -0.009859836660325527, 0.004081165883690119, 0.026717817410826683, -0.004025883506983519, 0.02384311519563198, 0.04170268774032593, -0.03262331336736679, -0.0015568591188639402, 0.005186820402741432, -0.02273746021091938, -0.006373773328959942, 0.005499005317687988, -0.003212901996448636]

近傍探索について

公式サイト

facebookresearch/faiss

<https://github.com/facebookresearch/faiss>

埋め込みの主な使用例は近傍探索である

入力テキストの意味の近いものを対象のテキストから探すタスクのことが近傍探索である

本章ではFacebookが開発したオープンソースのベクトルデータベース Faiss を使って近傍探索を実行する

モデル: text-embedding-ada-002

前準備:

Faiss の GPU版を使うため、GPUを有効化する

ノートブック→メニュー→編集→ノートブックの設定→GPU

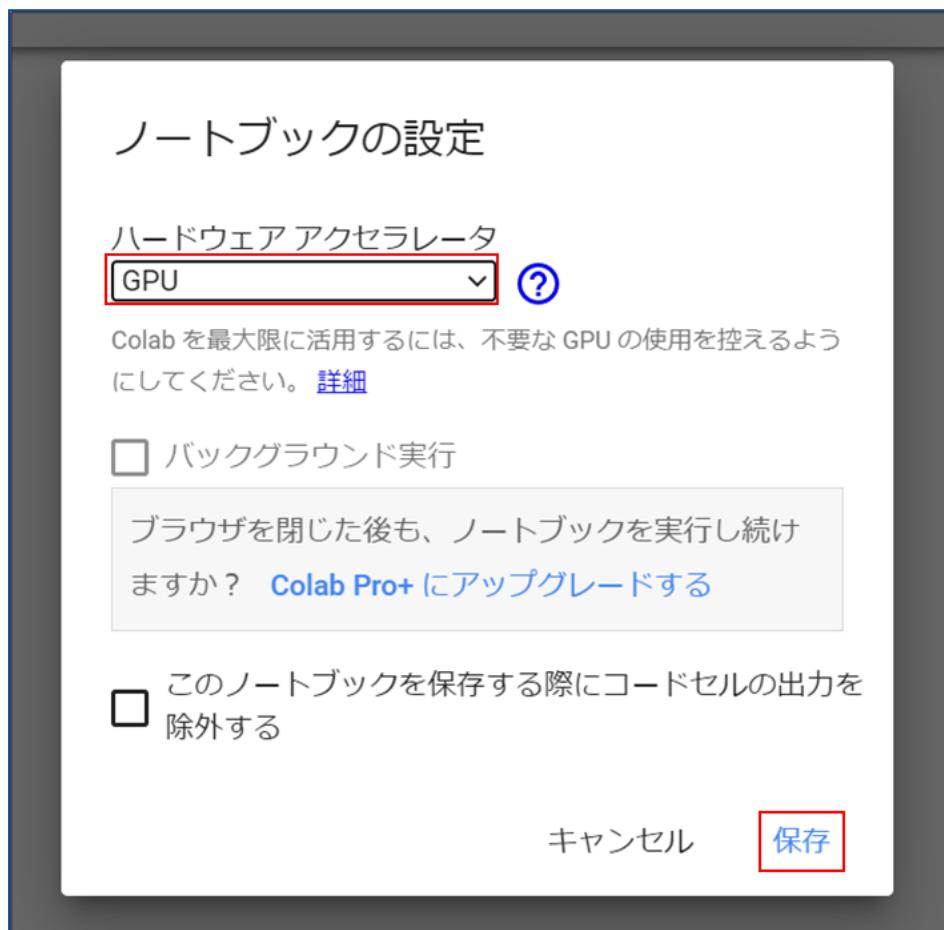


図4.11 出所: Skill Up AIより. 公式Webサイト |Colabの環境をGPUに設定するため、「ハードウェア アクセラレータ」を「GPU」に変更して「保存」をする画面

近傍探索アルゴリズムのIndex

| インデックス | 説明 |
|----------------------------------|---------------------------|
| IndexFlatL2(L2ノルム)
基本的なインデックス | ユークリッド距離を使用してベクトルの距離を計算する |
| IndexFlatIP(コサイン類似度) | 内積を使用してベクトル間の類似度を計算する |

| インデックス | 説明 |
|-------------------------|--------------------------------------|
| IndexIVFFlat(高速化アルゴリズム) | 高次元ベクトルをクラスタリングすることによってベクトルの検索を高速化する |

index.search()の引数

| 引数 | 説明 |
|-----------|-------------------|
| in_embeds | 検索したいベクトルのNumpy配列 |
| k | 返される最近傍ベクトルの数 |

index.search()の戻り値

| 戻り値 | 説明 |
|-----|------------------|
| D | 最も近いベクトルとの距離 |
| I | 最も近いベクトルとのインデックス |

テキスト生成と同じ前準備をする:

1. Faissパッケージのインストール

```
!pip install faiss-gpu
```

2. 環境変数の準備

```
import openai
import numpy as np
```

3. 入力テキストの準備

```
in_text = "今日は雨振らなくてよかったです"
```

4. パラメーターを定義

```
response = openai.Embedding.create(
```

5. 変換元の定義

```
input=in_text,
```

6. モデルを定義

```
model="text-embedding-ada-002"
)
```

7.入力テキストの埋め込み生成

```
in_embeds = [record["embedding"] for record in response["data"]]
in_embeds = np.array(in_embeds).astype("float32")
```

8.近傍探索出力用対象テキストの準備

```
target_texts = [
    "好きな食べ物は何ですか？",
    "どこにお住まいですか？",
    "朝の電車は混みますね",
    "今日は良いお天気ですね",
    "最近景気悪いですね"]
```

9.出力テキスト用パラメーター、変換元、モデルを定義

```
response = openai.Embedding.create(
    input=target_texts,
    model="text-embedding-ada-002"
)
```

10.対象テキストの埋め込み生成:

```
target_embeds = [record["embedding"] for record in response["data"]]
target_embeds = np.array(target_embeds).astype("float32")
```

11.Faissのインデックス生成

```
import faiss
index = faiss.IndexFlatL2(1536)
```

12.対象テキストをインデックスに追加

```
index.add(target_embeds)
```

13.近傍探索の実行

```
D, I = index.search(in_embeds, 1)
```

14.表示命令

```
print(D)
print(I)
print(target_texts[I[0][0]])
```

8.表示結果

入力テキストの意味の近いものを対象のテキストから探し、近傍探索した結果が表示される

[[0.17803922]]

[3]

今日は良いお天気ですね

```
In [ ]: import openai
import numpy as np

# 入力テキストの埋め込み生成
in_text = "今日は雨振らなくてよかった"
response = openai.Embedding.create(
    input=in_text,
    model="text-embedding-ada-002"
)
in_embeds = [record["embedding"] for record in response["data"]]
in_embeds = np.array(in_embeds).astype("float32")

# 対象テキストの埋め込み生成
target_texts = [
    "好きな食べ物は何ですか?",
    "どこにお住まいですか?",
    "朝の電車は混みますね",
    "今日は良いお天気ですね",
    "最近景気悪いですね"]
response = openai.Embedding.create(
    input=target_texts,
    model="text-embedding-ada-002"
)
target_embeds = [record["embedding"] for record in response["data"]]
target_embeds = np.array(target_embeds).astype("float32")
```

```
In [ ]: import faiss  
  
# Faissのインデックス生成  
index = faiss.IndexFlatL2(1536)
```

```
In [ ]: # 対象テキストをインデックスに追加  
index.add(target_embeds)
```

```
In [ ]: # 近傍探索の実行  
D, I = index.search(in_embeds, 1)
```

```
# 確認
print(D)
print(I)
print(target_texts[I[0][0]])
```

```
[[0.17803922]]
[[3]]
今日は良いお天気ですね
```

```
In [ ]: !pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.42.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

ファインチューニング

ファインチューニングとは、既に学習済みの言語モデルを、特定のタスクに最適化することである

公式サイト

Fine-tuning -OpenAI API

<https://platform.openai.com/docs/guides/fine-tuning>

ファインチューニングは事前学習済みモデルをベースに、個別タスクに合わせて行う追加学習である

利点としてはプロンプトで例を提供する必要がなくなり、結果としてトークン数が節約され、低遅延のリクエストが可能になる

ファインチューニング可能なモデル

- Davinci
- Curie
- Babbage

- Ada

Fine-tuning modelsの料金（学習）

| モデル | 料金（ドル） | 円（1ドル=130円レートで計算） |
|---------|------------------|-------------------|
| Davinci | 0.0300 ドル/1Kトークン | 3.900 |
| Curie | 0.0030 ドル/1Kトークン | 0.390 |
| Babbage | 0.0006 ドル/1Kトークン | 0.078 |
| Ada | 0.0004 ドル/1Kトークン | 0.052 |

Fine-tuning modelsの料金（使用）

| モデル | 料金（ドル） | 円（1ドル=130円レートで計算） |
|---------|------------------|-------------------|
| Davinci | 0.1200 ドル/1Kトークン | 15.600 |
| Curie | 0.0120 ドル/1Kトークン | 1.560 |
| Babbage | 0.0024 ドル/1Kトークン | 0.312 |
| Ada | 0.0016 ドル/1Kトークン | 0.208 |

前準備:

パッケージのインストール

```
!pip install openai
!pip install --upgrade openai
```

環境変数の準備

```
import os
os.environ["OPENAI_API_KEY"] = "Set your OpenAI API
key"
```

事前にT4:GPUを有効化する

ノートブック→メニュー→編集→ノートブックの設定→GPU

使われるモデル名を確認

例 モデルID :davinci

```
In [ ]: # パッケージのインストール  
!pip install openai  
!pip install --upgrade openai
```

```
Collecting openai
  Downloading openai-0.27.8-py3-none-any.whl (73 kB)
    73.6/73.6 kB 1.3 MB/s eta 0: 00:00
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from openai) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai) (4.65.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from openai) (3.8.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2023.7.22)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (4.0.2)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.3.1)
Installing collected packages: openai
Successfully installed openai-0.27.8
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages (0.27.8)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-packages (from openai) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai) (4.65.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from openai) (3.8.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai) (2023.7.22)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (4.0.2)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->openai) (1.3.1)
```

```
In [ ]: # 環境変数の準備 #Set your OpenAI API key  
import os  
os.environ["OPENAI_API_KEY"] = "your_API_key"
```

学習データの準備:

<https://www.borndigital.co.jp/book/%E3%81%AE%E6%83%85%E5%A0%B1%E3%81%A>

1. 日本語の会話学習済のエクセル・データベース「つくよみちゃん会話AI育成計画.xlsx」ファイルを使用してファインチューニングを行う

「つくよみちゃん会話AI育成計画.xlsx」のデータ形式について:

会話生成のテキストデータはプロンプト'prompt'とコンプリーション'completion'というペアで構成されている

```
names=['prompt','completion'],
```

2. ファイルのダウンロード先

<https://tyc.rei-yumesaki.net/material/kaiwa-ai/>

3. ダウンロードファイルのファイル名をColabで対応可能なファイルタイプへ変更し、PCに保存

```
tsukuyomi.csv
```

4. このファイルをColabのフォルダにアップロード

フォルダマークをクリック-> ツールバーの↑を押す -> PCに保存したファイルを選択 -> アップロード完了

5. 以下のコードでコードセルにデータベースを追加し、データセットの準備が完了

```
import pandas as pd  
  
df = pd.read_csv(  
    './tsukuyomi.csv',  
    usecols=[1,2],  
    names=['prompt','completion'],  
    skiprows=2)
```

このコマンドで names=['prompt','completion']データが tsukuyomi.jsonl 形式へ変換される

```
df.to_json(  
    "tsukuyomi.jsonl",  
    orient='records',  
    lines=True,  
    force_ascii=False  
)
```

データを表示してみる

df.head() を追加して'prompt','completion'の内容を確認する

```
In [ ]: import pandas as pd

# データセットの準備

df = pd.read_csv(
    './tsukuyomi.csv',
    usecols=[1,2],
    names=['prompt', 'completion'],
    skiprows=2)
df.head()
```

| | prompt | completion |
|---|-----------------|---------------------------------------|
| 0 | お腹が鳴る | 何か召し上がりりますか？ |
| 1 | だるい | それは心配です。私にできることがあれば、何でもお申し付けください。 |
| 2 | 眠れない | 何も心配する必要はありません。私がついています。ずっと、ずっと、一緒です。 |
| 3 | 押すなよ、絶対押す
なよ | はい、絶対押しません！ |
| 4 | 見せられないよ | 目をつぶって待っています。 |

.csv -> .jsonl へデータ変換

```
In [ ]: import pandas as pd

# データセットの準備

df = pd.read_csv(
    './tsukuyomi.csv',
    usecols=[1,2],
    names=['prompt', 'completion'],
    skiprows=2)
df.to_json(
    "tsukuyomi.jsonl",
    orient='records',
    lines=True,
    force_ascii=False
)
```

6. データセットの検証を行う

-f:検証ファイル

-q: ユーザー入力なしに、全提案を受け入れて自動修正

```
!openai tools fine_tunes.prepare_data \
-f tsukuyomi.jsonl \
-q
```

7. 検証結果の表示例（全文）：

Analyzing...

```
- Your file contains 998 prompt-completion pairs
<!-- - There are 530 duplicated prompt-completion sets. These
are rows: [438, 439, 470, 471, 472, 473, 474, 475, 476, 477,
478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489,
490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501,
502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513,
514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525,
526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537,
538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549,
550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561,
562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573,
574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585,
586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597,
598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609,
610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621,
622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633,
634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645,
646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657,
658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669,
670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681,
682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693,
694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705,
706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717,
718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729,
730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741,
742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753,
754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765,
766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777,
778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789,
790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801,
802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813,
814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825,
826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837,
838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849,
850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861,
862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873,
874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885,
886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897,
898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909,
910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921,
922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933,
934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945,
```

946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997] -->

– Your data does not contain a common separator at the end of your prompts. Having a separator string appended to the end of the prompt makes it clearer to the fine-tuned model where the completion should begin. See

<https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset> for more detail and examples. If you intend to do open-ended generation, then you should leave the prompts empty

– Your data does not contain a common ending at the end of your completions. Having a common ending string appended to the end of the completion makes it clearer to the fine-tuned model where the completion should end. See

<https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset> for more detail and examples.

– The completion should start with a whitespace character (` `). This tends to produce better results due to the tokenization we use. See

<https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset> for more details

Based on the analysis we will perform the following actions:

– [Recommended] Remove 530 duplicate rows [Y/n]: Y

– [Recommended] Add a suffix separator ` ->` to all prompts [Y/n]: Y

/usr/local/lib/python3.10/dist-

packages/openai/validators.py:226: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a
DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:

https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

x["prompt"] += suffix

– [Recommended] Add a suffix ending `\\n` to all completions [Y/n]: Y

/usr/local/lib/python3.10/dist-

packages/openai/validators.py:382: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a
DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:

https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

x["completion"] += suffix

– [Recommended] Add a whitespace character to the beginning of the completion [Y/n]: Y

/usr/local/lib/python3.10/dist-

packages/openai/validators.py:425: SettingWithCopyWarning:

```
A value is trying to be set on a copy of a slice from a
DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
```

```
https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
```

```
x["completion"] = x["completion"].apply(
```

```
Your data will be written to a new JSONL file. Proceed [Y/n]:
Y
```

```
Wrote modified file to `tsukuyomi_prepared.jsonl`
Feel free to take a look!
```

```
Now use that file when fine-tuning:
```

```
> openai api fine_tunes.create -t "tsukuyomi_prepared.jsonl"
```

```
After you've fine-tuned a model, remember that your prompt
has to end with the indicator string `->` for the model to
start generating completions, rather than continuing with the
prompt. Make sure to include `stop=["\n"]` so that the
generated texts ends at the expected place.
```

```
Once your model starts training, it'll approximately take
26.18 minutes to train a `curie` model, and less for `ada` and
`babbage`. Queue will approximately take half an hour per job
ahead of you.
```

8. 検証したデータセットは以下のファイル名でColabフォルダに出力される

このファイルは項6の修正によって会話の文章が改行されたものになる

tsukuyomi_prepared.jsonl

```
In [ ]: # データセットの検証
!openai tools fine_tunes.prepare_data \
-f tsukuyomi.jsonl \
-q
```

```
/bin/bash: line 1: openai: command not found
```

ファインチューニングの実行

```
!openai api fine_tunes.create \
-t "tsukuyomi_prepared.jsonl" \
-m <モデルID>
```

以下のファイルIDが表示される

IDが表示結果の3と4行目に表示される↓

```
=====
Found potentially duplicated files with name
'tsukuyomi_prepared.jsonl', purpose 'fine-tune' and size
71445 bytes
file-8DX4qzZY0LB8oN7rUiPD08Qv
file-AeCPJ6NExDkjWwVBYuV0XtxQ
Enter file ID to reuse an already uploaded file, or an
empty string to upload this file anyway:
```

ファイルIDをメモしておいてください

```
file-8DX4qzZY0LB8oN7rUiPD08Qv
file-AeCPJ6NExDkjWwVBYuV0XtxQ
```

後でファイルの再アップロード時にファイルのID入力が求められる

または

ファイルIDの入力が求められるときに空のエンターを押してもOK

ファインチューニングの実行結果の例(全文) :

```
=====
Found potentially duplicated files with name
'tsukuyomi_prepared.jsonl', purpose 'fine-tune' and size
71445 bytes
file-8DX4qzZY0LB8oN7rUiPD08Qv
file-AeCPJ6NExDkjWwVBYuV0XtxQ
Enter file ID to reuse an already uploaded file, or an
empty string to upload this file anyway:
Upload progress: 100% 71.4k/71.4k [00:00<00:00,
81.4Mit/s]
Uploaded file from tsukuyomi_prepared.jsonl: file-
huEnUYUWmlVLkQY1NV4Gke0i
Created fine-tune: ft-5jZl5NYoXQD3JhJNfnUNPyey
Streaming events until fine-tuning is complete...

(Ctrl-C will interrupt the stream, but not cancel the
fine-tune)
[2023-08-09 06:44:31] Created fine-tune: ft-
5jZl5NYoXQD3JhJNfnUNPyey

Stream interrupted (client disconnected).
To resume the stream, run:

    openai api fine_tunes.follow -i ft-
5jZl5NYoXQD3JhJNfnUNPyey
```

```
In [ ]: # ファインチューニングの実行
!openai api fine_tunes.create -t "tsukuyomi_prepared.jsonl" -m davinci
```

```
Found potentially duplicated files with name 'tsukuyomi_prepared.jsonl', purpose 'fine-tune' and size 71445 bytes
file-huEnUYUWmlVLkQY1NV4Gke0i
file-8DX4qzZY0LB8oN7rUiPD08Qv
file-AeCPJ6NExDkjWWVBYuVOXtxQ
file-urmrjFkx9XWXfAanFIkMnZf
file-n0MwoKl5ib8C51dM3745BHxH
file-z5FoCU2eM3agGtiR0CfLFMxO
file-VNalG599hLl8Yi1sjIr9yQGF
Enter file ID to reuse an already uploaded file, or an empty string to upload this file anyway:
Upload progress: 100% 71.4k/71.4k [00:00<00:00, 78.7Mit/s]
Uploaded file from tsukuyomi_prepared.jsonl: file-1FmxHg6KV2fQiFLqjjA2voz6
Created fine-tune: ft-6Gny0lfEQeuhTaYvPotGBftM
Streaming events until fine-tuning is complete...

(Ctrl-C will interrupt the stream, but not cancel the fine-tune)
[2023-08-10 04:59:55] Created fine-tune: ft-6Gny0lfEQeuhTaYvPotGBftM
```

Stream interrupted (client disconnected).
To resume the stream, run:

```
openai api fine_tunes.follow -i ft-6Gny0lfEQeuhTaYvPotGBftM
```

```
In [ ]: # ストリーム再開
!openai api fine_tunes.follow -i ft-6Gny0lfEQeuhTaYvPotGBftM
```

```
[2023-08-10 04:59:55] Created fine-tune: ft-6Gny0lfEQeuhTaYvPotGBftM
```

Stream interrupted (client disconnected).
To resume the stream, run:

```
openai api fine_tunes.follow -i ft-6Gny0lfEQeuhTaYvPotGBftM
```

```
In [ ]: # フайнチューニングのジョブの確認
!openai api fine_tunes.get -i ft-6Gny0lfEQeuhTaYvPotGBftM
```

```
{  
    "object": "fine-tune",  
    "id": "ft-3P2hQ2N1fADysEKSs7IvZ0uy",  
    "hyperparams": {  
        "n_epochs": 4,  
        "batch_size": null,  
        "prompt_loss_weight": 0.01,  
        "learning_rate_multiplier": null  
    },  
    "organization_id": "org-NjF1gp1qnlpNzEmqp6EJAW3q",  
    "model": "davinci",  
    "training_files": [  
        {  
            "object": "file",  
            "id": "file-AeCPJ6NExDkjWWVBYuV0XtxQ",  
            "purpose": "fine-tune",  
            "filename": "tsukuyomi_prepared.jsonl",  
            "bytes": 71445,  
            "created_at": 1691563371,  
            "status": "processed",  
            "status_details": null  
        }  
    ],  
    "validation_files": [],  
    "result_files": [],  
    "created_at": 1691633271,  
    "updated_at": 1691633271,  
    "status": "pending",  
    "fine_tuned_model": null,  
    "events": [  
        {  
            "object": "fine-tune-event",  
            "level": "info",  
            "message": "Created fine-tune: ft-3P2hQ2N1fADysEKSs7IvZ0uy",  
            "created_at": 1691633271  
        }  
    ]  
}
```

usage: openai api fine_tunes.get [-h] -i ID openai api fine_tunes.get: error: argument
-i/--id: expected one argument

```
In [ ]: # ジョブキャンセル  
!openai api fine_tunes.cancel -i ft-6Gny0lfEQeuhTaYvPotGBftM
```

```
{
  "object": "fine-tune",
  "id": "ft-6Gny0lfEQeuhTaYvPotGBftM",
  "hyperparams": {
    "n_epochs": 4,
    "batch_size": null,
    "prompt_loss_weight": 0.01,
    "learning_rate_multiplier": null
  },
  "organization_id": "org-NjF1gp1qnlpNzEmqp6EJAW3q",
  "model": "davinci",
  "training_files": [
    {
      "object": "file",
      "id": "file-1FmxHg6KV2fQiFLqjjA2voz6",
      "purpose": "fine-tune",
      "filename": "tsukuyomi_prepared.jsonl",
      "bytes": 71445,
      "created_at": 1691643595,
      "status": "processed",
      "status_details": null
    }
  ],
  "validation_files": [],
  "result_files": [],
  "created_at": 1691643595,
  "updated_at": 1691643784,
  "status": "cancelled",
  "fine_tuned_model": null,
  "events": [
    {
      "object": "fine-tune-event",
      "level": "info",
      "message": "Created fine-tune: ft-6Gny0lfEQeuhTaYvPotGBftM",
      "created_at": 1691643595
    },
    {
      "object": "fine-tune-event",
      "level": "info",
      "message": "Fine-tune cancelled",
      "created_at": 1691643784
    }
  ]
}
```

推論の実行

```
In [ ]: import openai

# 推論の実行
prompt="好きな食べ物は何?->"
response = openai.Completion.create(
    model="davinci",
    prompt=prompt,
    max_tokens=100,
    stop="\n")
print(response["choices"][0]["text"])
```

かぼちゃ!

ファインチューニングモデルの一覧取得と削除

```
In [ ]: # ファインチューニングモデル一覧確認  
!openai api fine_tunes.list
```

```
{
  "object": "list",
  "data": [
    {
      "object": "fine-tune",
      "id": "ft-LwhVixx5AX3ZCE0hnUokIPxy",
      "hyperparams": {
        "n_epochs": 4,
        "batch_size": 1,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": 0.1
      },
      "organization_id": "org-NjF1gp1qnlpNzEmqp6EJAW3q",
      "model": "davinci",
      "training_files": [
        {
          "object": "file",
          "id": "file-rWnIimRLzZJVSKcUNMo8bY0C",
          "purpose": "fine-tune",
          "filename": "tsukuyomi_prepared.jsonl",
          "bytes": 71173,
          "created_at": 1686129512,
          "status": "processed",
          "status_details": null
        }
      ],
      "validation_files": [],
      "result_files": [
        {
          "object": "file",
          "id": "file-7LD1GSLCFCyB30J8MUCMRTNB",
          "purpose": "fine-tune-results",
          "filename": "compiled_results.csv",
          "bytes": 99676,
          "created_at": 1686130553,
          "status": "processed",
          "status_details": null
        }
      ],
      "created_at": 1686129512,
      "updated_at": 1686130553,
      "status": "succeeded",
      "fine_tuned_model": "davinci:ft-shinshu-university-2023-06-07-09-35-51"
    },
    {
      "object": "fine-tune",
      "id": "ft-eRQNLLuAAs4YgVQwsmENg9iL",
      "hyperparams": {
        "n_epochs": 4,
        "batch_size": null,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": null
      },
      "organization_id": "org-NjF1gp1qnlpNzEmqp6EJAW3q",
      "model": "davinci",
      "training_files": [
        {
          "object": "file",
          "id": "file-8DX4qzZY0LB8oN7rUiPD08Qv",

```

```
        "purpose": "fine-tune",
        "filename": "tsukuyomi_prepared.jsonl",
        "bytes": 71445,
        "created_at": 1691560952,
        "status": "processed",
        "status_details": null
    }
],
"validation_files": [],
"result_files": [],
"created_at": 1691560952,
"updated_at": 1691560952,
"status": "pending",
"fine_tuned_model": null
},
{
    "object": "fine-tune",
    "id": "ft-gM8s9TYpr8jyyH0XSwWLSUuv",
    "hyperparams": {
        "n_epochs": 4,
        "batch_size": null,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": null
    },
    "organization_id": "org-NjF1gp1qnlpNzEmqp6EJAW3q",
    "model": "davinci",
    "training_files": [
        {
            "object": "file",
            "id": "file-AeCPJ6NExDkjWWVBYuVOXtxQ",
            "purpose": "fine-tune",
            "filename": "tsukuyomi_prepared.jsonl",
            "bytes": 71445,
            "created_at": 1691563371,
            "status": "processed",
            "status_details": null
        }
    ],
    "validation_files": [],
    "result_files": [],
    "created_at": 1691563371,
    "updated_at": 1691563371,
    "status": "pending",
    "fine_tuned_model": null
},
{
    "object": "fine-tune",
    "id": "ft-5jZl5NYoXQD3JhJNfnUNPyey",
    "hyperparams": {
        "n_epochs": 4,
        "batch_size": null,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": null
    },
    "organization_id": "org-NjF1gp1qnlpNzEmqp6EJAW3q",
    "model": "davinci",
    "training_files": [
        {
            "object": "file",
            "id": "file-huEnUYUWmlVLkQY1NV4Gke0i",
            "purpose": "fine-tune",
            "filename": "tsukuyomi_prepared.jsonl",
            "bytes": 71445,
            "created_at": 1691563371,
            "status": "processed",
            "status_details": null
        }
    ],
    "validation_files": [],
    "result_files": [],
    "created_at": 1691563371,
    "updated_at": 1691563371,
    "status": "pending",
    "fine_tuned_model": null
}
```

```
        "purpose": "fine-tune",
        "filename": "tsukuyomi_prepared.jsonl",
        "bytes": 71445,
        "created_at": 1691563471,
        "status": "processed",
        "status_details": null
    }
],
"validation_files": [],
"result_files": [],
"created_at": 1691563471,
"updated_at": 1691563471,
"status": "pending",
"fine_tuned_model": null
},
{
    "object": "fine-tune",
    "id": "ft-3P2hQ2N1fADysEKSs7IvZ0uy",
    "hyperparams": {
        "n_epochs": 4,
        "batch_size": null,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": null
    },
    "organization_id": "org-NjF1gp1qnlpNzEmqp6EJAW3q",
    "model": "davinci",
    "training_files": [
        {
            "object": "file",
            "id": "file-AeCPJ6NExDkjWWVBYuVOXtxQ",
            "purpose": "fine-tune",
            "filename": "tsukuyomi_prepared.jsonl",
            "bytes": 71445,
            "created_at": 1691563371,
            "status": "processed",
            "status_details": null
        }
    ],
    "validation_files": [],
    "result_files": [],
    "created_at": 1691633271,
    "updated_at": 1691633271,
    "status": "pending",
    "fine_tuned_model": null
}
]
}

{
    "object": "list",
    "data": [
        {
            "object": "fine-tune",
            "id": "ft-LwhVixx5AX3ZCE0hnUokIPxy",
            "hyperparams": {
                "n_epochs": 4,
                "batch_size": 1,
                "prompt_loss_weight": 0.01,
                "learning_rate_multiplier": null
            }
        }
    ]
}
```

```
        "learning_rate_multiplier": 0.1
    },
    "organization_id": "org-
NjF1gp1qnlpNzEmqp6EJAw3q",
    "model": "davinci",
    "training_files": [
        {
            "object": "file",
            "id": "file-rWnIimRLzZJVSKcUNMo8bY0C",
            "purpose": "fine-tune",
            "filename": "tsukuyomi_prepared.jsonl",
            "bytes": 71173,
            "created_at": 1686129512,
            "status": "processed",
            "status_details": null
        }
    ],
    "validation_files": [],
    "result_files": [
        {
            "object": "file",
            "id": "file-7LD1GSLCFCyB30J8MUCMRTNB",
            "purpose": "fine-tune-results",
            "filename": "compiled_results.csv",
            "bytes": 99676,
            "created_at": 1686130553,
            "status": "processed",
            "status_details": null
        }
    ],
    "created_at": 1686129512,
    "updated_at": 1686130553,
    "status": "succeeded",
    "fine_tuned_model": "davinci:ft-shinshu-
university-2023-06-07-09-35-51"
},
{
    "object": "fine-tune",
    "id": "ft-eRQNLLuAAs4YgVQwsmENg9iL",
    "hyperparams": {
        "n_epochs": 4,
        "batch_size": null,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": null
    },
    "organization_id": "org-
NjF1gp1qnlpNzEmqp6EJAw3q",
    "model": "davinci",
    "training_files": [
        {
            "object": "file",
            "id": "file-8DX4qzZY0LB8oN7rUiPD08Qv",
            "purpose": "fine-tune",
            "filename": "tsukuyomi_prepared.jsonl",
            "bytes": 71445,
            "created_at": 1691560952,
```

```
        "status": "processed",
        "status_details": null
    }
],
"validation_files": [],
"result_files": [],
"created_at": 1691560952,
"updated_at": 1691560952,
"status": "pending",
"fine_tuned_model": null
},
{
    "object": "fine-tune",
    "id": "ft-gM8s9TYpr8jyyH0XSwWLSUUv",
    "hyperparams": {
        "n_epochs": 4,
        "batch_size": null,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": null
    },
    "organization_id": "org-
NjF1gp1qnlpNzEmqp6EJAW3q",
    "model": "davinci",
    "training_files": [
        {
            "object": "file",
            "id": "file-AeCPJ6NExDkjWWVBYuVOXtxQ",
            "purpose": "fine-tune",
            "filename": "tsukuyomi_prepared.jsonl",
            "bytes": 71445,
            "created_at": 1691563371,
            "status": "processed",
            "status_details": null
        }
    ],
    "validation_files": [],
    "result_files": [],
    "created_at": 1691563371,
    "updated_at": 1691563371,
    "status": "pending",
    "fine_tuned_model": null
},
{
    "object": "fine-tune",
    "id": "ft-5jZl5NYoXQD3JhJNfnUNPyey",
    "hyperparams": {
        "n_epochs": 4,
        "batch_size": null,
        "prompt_loss_weight": 0.01,
        "learning_rate_multiplier": null
    },
    "organization_id": "org-
NjF1gp1qnlpNzEmqp6EJAW3q",
    "model": "davinci",
    "training_files": [
        {
```

```

        "object": "file",
        "id": "file-huEnUYUWmlVLkQY1NV4Gke0i",
        "purpose": "fine-tune",
        "filename": "tsukuyomi_prepared.jsonl",
        "bytes": 71445,
        "created_at": 1691563471,
        "status": "processed",
        "status_details": null
    }
],
"validation_files": [],
"result_files": [],
"created_at": 1691563471,
"updated_at": 1691563471,
"status": "pending",
"fine_tuned_model": null
}
]
}

```

```
In [ ]: # # ファインチューニングモデルの削除
# !openai api models.delete -i <モデルID>
```

モデレーション

モデレーションはユーザー入力やLLMの出力に、暴力や自傷やヘイトやセクシャルなどの問題発言が含まれていないかどうか（OpenAIのコンテンツポリシーに準拠するかどうか）を判定するAPIである

Usage policies -OpenAI API

<https://openai.com/policies/usage-policies>

Moderation -OpenAI API

<https://platform.openai.com/docs/guides/moderation>

モデレーションのカテゴリー

- hate:人種、性別、民族性、宗教、国籍、性的指向、障害の状態、カーストに基づくに表現、扇動、または助長するコンテンツ
- hate/threatening:対象グループに対する暴力や深刻な危害を含むヘイトコンテンツ
- self-harm:自殺、切断、などの自傷行為を助長するコンテンツ
- sexual:性行為などの説明など、または性的なサービスを促進するコンテンツ
- sexual/minors:18歳未満の個人を含む性的コンテンツ
- violence:暴力を助長または美化するコンテンツ他人の苦しみを称賛するコンテンツ
- violence/graphic:死、暴力、または深刻な身体的損傷を極度に描写する暴力的なコンテンツなど

利用料金:

モデレーションはOpenAI APIの入出力に対して無料で利用できる

前準備:

テキスト生成のと同様に準備をする

モデレーションの利用について

モデル :

最も高性能な最新モデル

text-moderation-latest

最新のモデルよりわずかに古いモデル

text-moderation-stable

response:

OpenAIのコンテンツポリシー違反かどうか判定 (true/false)

レスポンスの判定が true の場合、flagged と返す

パラメーター:

```
response = openai.Moderation.create()
```

```
In [ ]: # パッケージのインポート  
!pip install openai
```

```
In [ ]: # 環境変数の準備  
import os  
os.environ["OPENAI_API_KEY"] = "<OpenAI_APIのAPIキー>"
```

```
In [ ]: import openai  
  
# モデレーションの利用  
response = openai.Moderation.create(  
    input="お前を殺す!"  
)  
print(response)
```

音声テスト変換

Whisper -OpenAI

Whisper -OpenAIについて

オープンソースの高品質な音声認識のAIモデル「whisper-large-v2」でテキストに変換する

機能：

音声を文字起こし

音声を英語に翻訳して文字起こし

ファイルサイズ: 最大25MB (mp3, mp4, mpeg, mpga, m4a, wav, webm)

Transformerのシーケンス対シーケンスモデルは、多言語音声認識、音声翻訳、対話言語識別、音声活動検出などのさまざまな音声処理タスクで訓練されている

これらのタスクは、デコーダーによって予測されるトークンのシーケンスとして共同で表現されており、単一のモデルが従来の音声処理パイプラインの多くのステージを置き換える

マルチタスクのトレーニングフォーマットでは、タスク指示子や分類ターゲットとして機能する特別なトークンセットが使用されている

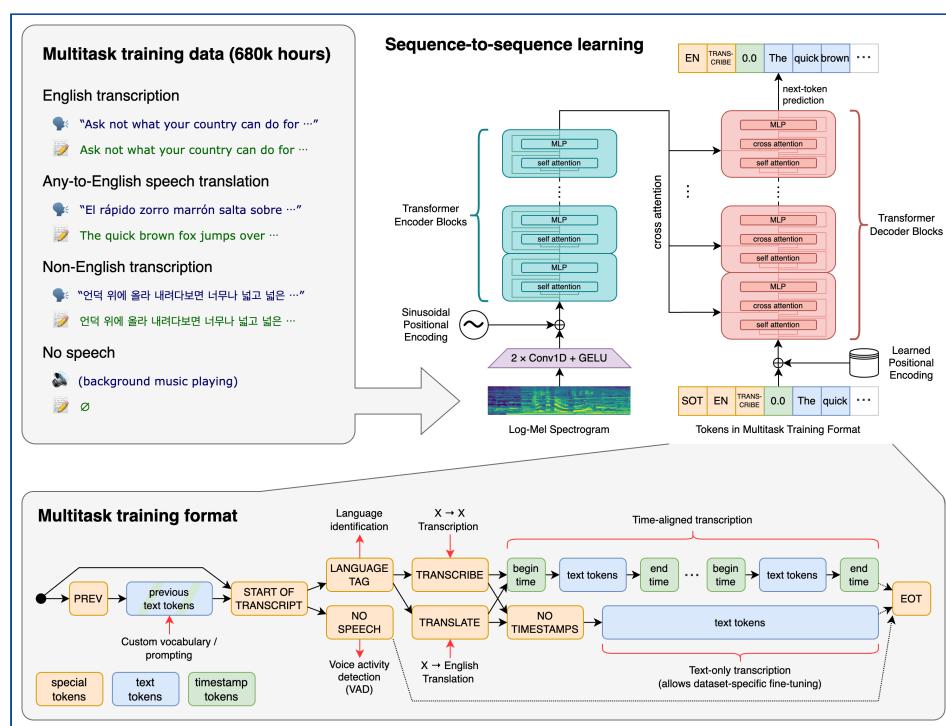


図4.12 出所: OpenAIより. 公式Webサイト |Whisperのトレーニング構造およびマルチタスクの構成

参考リポジトリサイト

<https://github.com/openai/whisper>

生成された言語別のエラー率

WER (Word Error Rate)

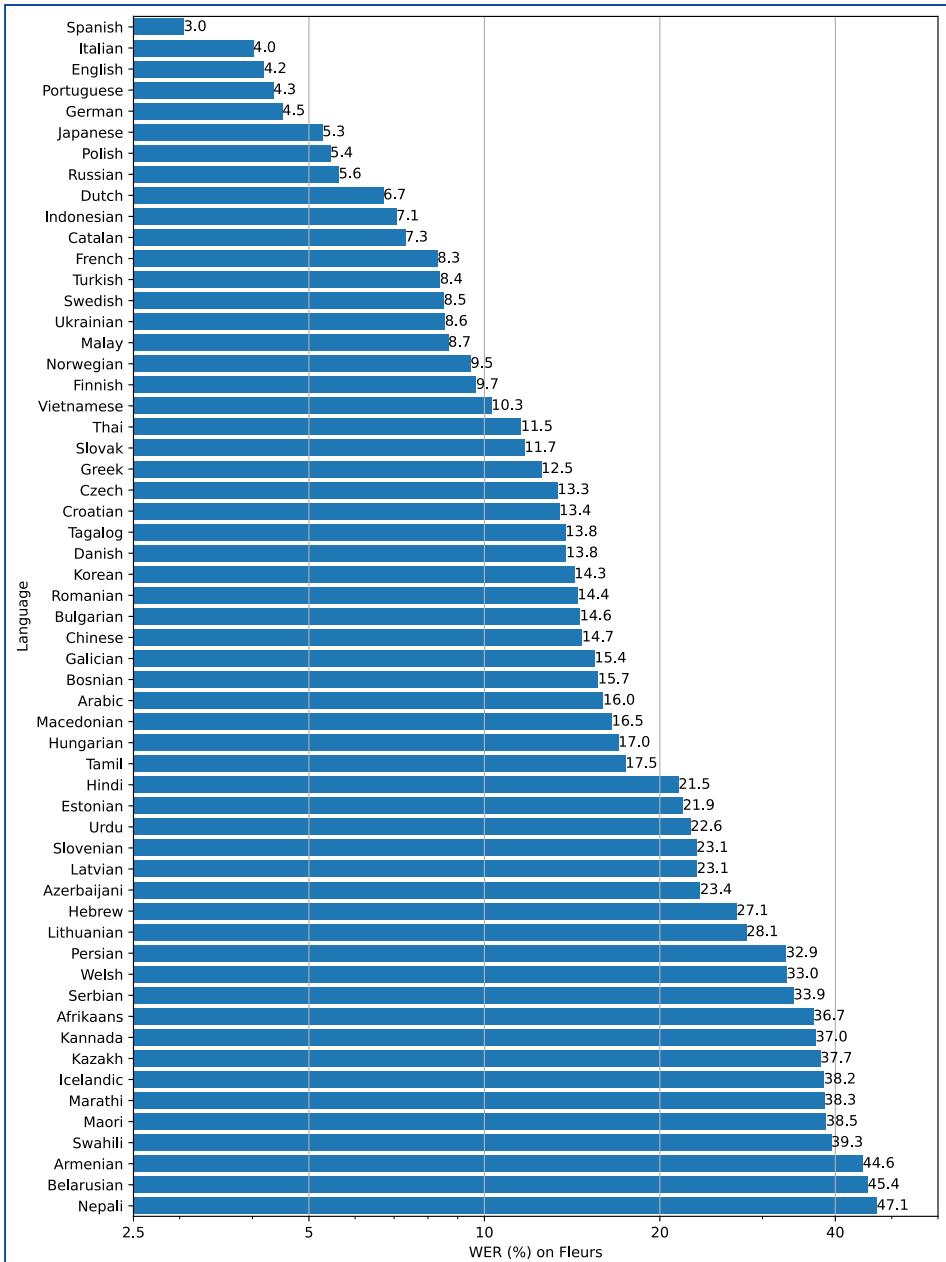


図4.13 出所: OpenAIより. 公式Webサイト |生成された言語別のエラー率を示すグラフ

利用料金

OpenAIの利用料金がかかる、音声の時間によって料金が変わる

0.006 ドル/1分 (秒単位で四捨五入)

前準備

テキスト生成の「OpenAI APIの前準備」をご参照ください

```
In [ ]: # パッケージのインポート
!pip install openai
```

```
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-pa  
ckages (0.27.9)  
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.1  
0/dist-packages (from openai) (2.31.0)  
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-pac  
kages (from openai) (4.66.1)  
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-p  
ackages (from openai) (3.8.5)  
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/  
python3.10/dist-packages (from requests>=2.20->openai) (3.2.0)  
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d  
ist-packages (from requests>=2.20->openai) (3.4)  
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python  
3.10/dist-packages (from requests>=2.20->openai) (2.0.4)  
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python  
3.10/dist-packages (from requests>=2.20->openai) (2023.7.22)  
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/  
dist-packages (from aiohttp->openai) (23.1.0)  
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/pytho  
n3.10/dist-packages (from aiohttp->openai) (6.0.4)  
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/l  
ib/python3.10/dist-packages (from aiohttp->openai) (4.0.3)  
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.1  
0/dist-packages (from aiohttp->openai) (1.9.2)  
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python  
3.10/dist-packages (from aiohttp->openai) (1.4.0)  
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.  
10/dist-packages (from aiohttp->openai) (1.3.1)
```

```
In [ ]: # 環境変数の準備  
import os  
os.environ["OPENAI_API_KEY"] = "<OpenAI_APIのAPIキー>"
```

音声文字起こし

音声文字起こしの流れ :

音声ファイルの作成

予めに音声を録音する

↓

録音したファイル(例:/content/レコーディング.m4a) をColabフォルダにアップロード
する

↓

コードセルに読み取る

```
audio_file= open("レコーディング.m4a", "rb")
```

↓

以下のコードで文字起こしをする

```
transcript = openai.Audio.transcribe("whisper-1",
audio_file)
```

↓

表示させる

```
print(transcript["text"])
```

↓

表示結果

Ah, hello. Good morning, my good sir. Would you like a cup of tea? Ah, hello.

```
In [ ]: #実行例
import openai

# 音声の文字起こし（オーディオファイルをカレントフォルダに置く）
audio_file= open("レコーディング.m4a", "rb")
transcript = openai.Audio.transcribe("whisper-1", audio_file)
print(transcript["text"])
```

Ah, hello. Good morning, my good sir. Would you like a cup of tea? Ah, hello.

openai.Audio.transcribe()のパラメーター

パラメーター	説明
file	文字起こす音声ファイル
model	モデルID（現在は「whisper-1」のみ）
prompt	モデルのスタイルをガイドするプロンプト
response_format	出力ファイルの形式 (json, text, srt, verbose_json, vtt, デフォルト:json)
temperature	創造的:0.8、答えがある場合: 0、

top_pと同時変更は非推奨(0~2、デフォルト: 1) |
| language | 入力音声の言語(「en」「ja」など)。これを付加しなくても自動的に判定されるが、指定した方が処理時間が短くなる |

音声を英語に翻訳して文字起こし

音声を英語に翻訳して文字起こしの流れ：

予めに音声を録音する

↓

録音したファイルをColabフォルダにアップロードする

↓

コードセルに読み取る

```
audio_file= open("/content/レコーディング.m4a", "rb")
```

↓

以下のコードで文字起こしをして翻訳する

```
transcript = openai.Audio.translate("whisper-1",
audio_file)
```

↓

表示させる

```
print(transcript["text"])
```

↓

表示結果

Ah, hello. Good morning, my good sir. Would you like a cup of tea? Ah, hello.の日本語翻訳版が表示される

注意：音声ファイルの品質が良くなければ翻訳されなかったり、エラーが発生したりします。

```
In [ ]: # 環境変数の準備
import os
os.environ["OPENAI_API_KEY"] = "<OpenAI_APIキー>"
```

```
In [ ]: #音声ファイルの翻訳実行例
import openai

# 音声を英語に翻訳して文字起こし（オーディオファイルをカレントフォルダに置く）
audio_file= open("レコーディング.m4a", "rb")
transcript = openai.Audio.translate("whisper-1", audio_file)
print(transcript["text"])
```

```
In [ ]: #表示結果がうまくいかなかったときは形式を確認するコード

try:
    transcript = openai.Audio.translate("whisper-1", audio_file)
    print(transcript["text"])
except openai.error.OpenAIError as e:
    print("An error occurred:", e)
```

```
An error occurred: Invalid file format. Supported formats: ['flac', 'm4a', 'mp3', 'mp4', 'mpeg', 'mpga', 'oga', 'ogg', 'wav', 'webm']
```

```
In [ ]: #入力ファイルの形式を変更するコード  
from pydub import AudioSegment  
  
input_file = "レコーディング.m4a"  
output_file = "output.wav"  
  
audio = AudioSegment.from_file(input_file, format="m4a")  
audio.export(output_file, format="wav")
```

```
Out[ ]: <_io.BufferedReader name='output.wav'>
```

より長いオーディオファイルの翻訳

ファイルのサイズを分割、圧縮することでより長いオーディオファイルの作成ができる

音声を文の途中で分割しないように注意してください

圧縮の流れ：

必要なモジュールのインポート:

```
from pydub import AudioSegment
```

オーディオファイルの読み込み:

```
song = AudioSegment.from_mp3("audio.mp3")
```

時間間隔の定義:

秒をミリ秒へ変換

```
ten_minutes = 10 * 60 * 1000
```

オーディオの一部を選択:

最初の 10 分 (ten_minutes 変数で定義されている) を選択する

```
first_10_minutes = song[:ten_minutes]
```

選択した一部をエクスポート:

```
first_10_minutes.export("audio.mp3", format="mp3")
```

```
In [ ]: !pip install pydub
```

```
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (0.25.1)
```

```
In [ ]: #ファイルの長さ圧縮実行例
```

```
from pydub import AudioSegment  
  
song = AudioSegment.from_mp3("audio.mp3")  
ten_minutes = 10 * 60 * 1000  
first_10_minutes = song[:ten_minutes]  
first_10_minutes.export("audio.mp3", format="mp3")
```

```
Out[ ]: <_io.BufferedReader name='/content/thirty-eight-greetings-29268.mp3'>
```

トークナイザー

tiktokenトークナイザーの特徴

LLMでは最小の単位のテキストを「トークン」という

これを分割することを「トークンナイザー」と呼ぶ

「tiktoken」は同等のオーポンソースのトークンナイザーより3~6倍高速な特徴がある

openai/titoken

<https://github.com/openai/tiktoken>

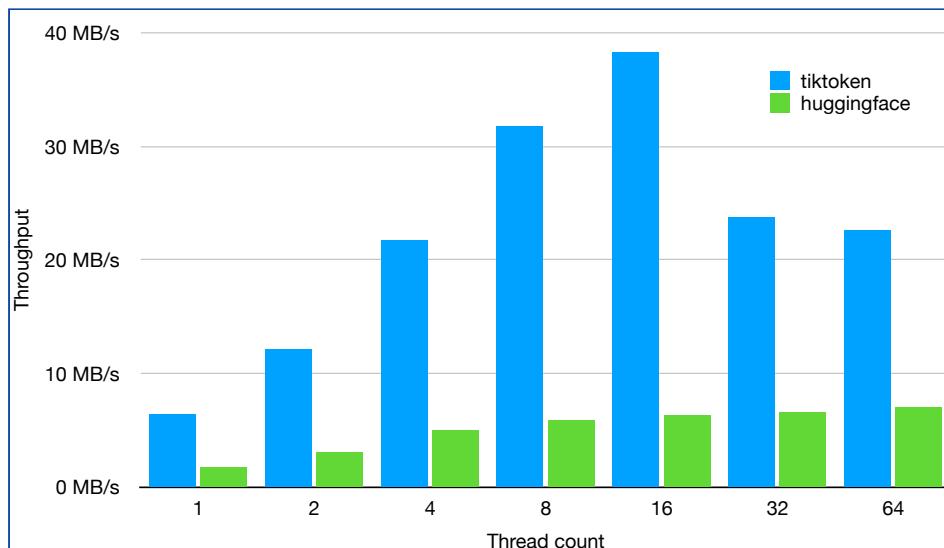


図4.14 出所: openaiより. 公式Webサイト |OpenAIのTiktokenの速度をHuggingfaceと比較した結果

最大トークン数:

LLMの呼び出しでは最大トークン数がモデルごとに決まっている

種別	モデル名	最大トークン数	エンコーディング
GPT-4	gpt-4	8192	cl100k_base
	gpt-4-0314	8192	cl100k_base
	gpt-4-32k	32768	cl100k_base

種別	モデル名	最大トークン数	エンコーディング
	gpt-4-32k-0314	32768	cl100k_base
GPT-3.5	gpt-3.5-turbo	4096	cl100k_base
	gpt-3.5-turbo-0301	4096	cl100k_base
	text-davinci-003	4097	p50k_base
	text-davinci-002	4097	p50k_base
GPT-3.0	text-curie-001	2049	r50k_base
	text-babbage-001	2049	r50k_base
	text-ada-001	2049	r50k_base
	davinci	2049	r50k_base
	curie	2049	r50k_base
	babbage	2049	r50k_base
	ada	2049	r50k_base
Embedding	text-embedding-ada-002	8192	cl100k_base

トークナイザーの利用

```
In [ ]: # tiktokenパッケージのインストール
!pip install tiktoken
```

```
In [ ]: import tiktoken

# エンコーディングの取得
enc = tiktoken.get_encoding("cl100k_base")
```

```
In [ ]: # エンコードの実行
tokens = enc.encode("Hello World!")
print(len(tokens))
print(tokens)
```

```
In [ ]: # デコードの実行
print(enc.decode(tokens))
```

```
In [ ]: # 分割したままでデコードを実行
print(enc.decode_tokens_bytes(tokens))
```

```
In [ ]:
```

日本語と英語のトークン数の比較

```
In [ ]: # エンコードの実行
tokens = enc.encode("こんにちは、世界!")
print(len(tokens))
print(tokens)
```

```
In [ ]: # デコードの実行  
print(enc.decode(tokens))
```

```
In [ ]: # 分割したままデコード  
def data2str(data):  
    try:  
        return data.decode('utf-8')  
    except UnicodeError:  
        return data  
print([data2str(data) for data in enc.decode_tokens_bytes(tokens)])
```

エンコーディングの確認

```
In [ ]: import tiktoken  
  
# エンコーディングの確認  
print(tiktoken.encoding_for_model("gpt-4"))
```