LLMA

LLaMA（Large Language Model Meta AI）は、Meta AI が2023年2月に発表した大規模言語モデル

4つのモデルサイズ: 7B（70億）、13B（130億）、30B（300億）、65B（650億）

70億パラメータから650億パラメータまで、さまざまなサイズのモデルが学習されたモデル

130億パラメータモデルがほとんどのNLPベンチマークにおいてGPT-3（1750億パラメータ）の性能を上回る

Meta は LLaMA のモデルのウェイトを非商用ライセンスで研究コミュニティに公開した

1.4兆個のトークンで学習したモデル

スタンフォード大学の基盤モデル研究センター（Center for Research on Foundation Models, CRFM）は、LLaMA の 70億パラメータ・モデルをファイン・チューニングした、Alpaca をリリース

Alpaca は OpenAI GPT-3.5シリーズの text-davinci-003モデルに相当する性能がある

図5.1. 出所:スタンフォード大学公式Webサイト |CRFMの70億パラメータLLaMAモデルーAlpaca

Alpaca LLaMA

チャットアプリケーションに特化したLLM

Alpacaは、より自然で流暢な会話を生成するために、コミュニケーションスキルや対話戦略を学習されている

Alpacaはより人間らしい回答を提供する

金融機関の金融商品についての言語処理を例とする

ある企業の株コードを入力し、決まったコード入力しただけで、市場の株価から始め市場分析まで可能である

LLMAをAPI経由での利用方法

Alpacaアカウントを登録

https://app.alpaca.markets/signup

メール、PW、ユーザー名入力で登録が可能

インストール

In [ ]:
```
!pip install alpaca-trade-api
```

```
Collecting alpaca-trade-api
  Downloading alpaca_trade_api-3.0.2-py3-none-any.whl (34 kB)
Requirement already satisfied: pandas>=0.18.1 in /usr/local/lib/python3.1
0/dist-packages (from alpaca-trade-api) (1.5.3)
Requirement already satisfied: numpy>=1.11.1 in /usr/local/lib/python3.10/
dist-packages (from alpaca-trade-api) (1.23.5)
Requirement already satisfied: requests<3,>2 in /usr/local/lib/python3.10/
dist-packages (from alpaca-trade-api) (2.31.0)
Collecting urllib3<2,>1.24 (from alpaca-trade-api)
  Downloading urllib3-1.26.16-py2.py3-none-any.whl (143 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.1/143.1 kB 6.5 MB/s eta
0:00:00
Requirement already satisfied: websocket-client<2,>=0.56.0 in /usr/local/l
ib/python3.10/dist-packages (from alpaca-trade-api) (1.6.1)
Collecting websockets<11,>=9.0 (from alpaca-trade-api)
  Downloading websockets-10.4-cp310-cp310-manylinux_2_5_x86_64.manylinux1_
x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (106 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.8/106.8 kB 16.0 MB/s eta
0:00:00
Collecting msgpack==1.0.3 (from alpaca-trade-api)
  Downloading msgpack-1.0.3-cp310-cp310-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl (323 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.7/323.7 kB 25.8 MB/s eta
0:00:00
Collecting aiohttp==3.8.2 (from alpaca-trade-api)
  Downloading aiohttp-3.8.2-cp310-cp310-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl (1.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 48.1 MB/s eta 0:0
0:00
Collecting PyYAML==6.0 (from alpaca-trade-api)
  Downloading PyYAML-6.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_6
4.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (682 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 682.2/682.2 kB 42.6 MB/s eta
0:00:00
Collecting deprecation==2.1.0 (from alpaca-trade-api)
  Downloading deprecation-2.1.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/
dist-packages (from aiohttp==3.8.2->alpaca-trade-api) (23.1.0)
Collecting charset-normalizer<3.0,>=2.0 (from aiohttp==3.8.2->alpaca-trade
-api)
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting multidict<6.0,>=4.5 (from aiohttp==3.8.2->alpaca-trade-api)
  Downloading multidict-5.2.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_
x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (175 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 175.1/175.1 kB 24.3 MB/s eta
0:00:00
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/l
ib/python3.10/dist-packages (from aiohttp==3.8.2->alpaca-trade-api) (4.0.
3)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.1
0/dist-packages (from aiohttp==3.8.2->alpaca-trade-api) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python
3.10/dist-packages (from aiohttp==3.8.2->alpaca-trade-api) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.
10/dist-packages (from aiohttp==3.8.2->alpaca-trade-api) (1.3.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist
-packages (from deprecation==2.1.0->alpaca-trade-api) (23.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/py
thon3.10/dist-packages (from pandas>=0.18.1->alpaca-trade-api) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/d
```

```
ist-packages (from pandas>=0.18.1->alpaca-trade-api) (2023.3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
ist-packages (from requests<3,>2->alpaca-trade-api) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.10/dist-packages (from requests<3,>2->alpaca-trade-api) (2023.7.22)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.1->pandas>=0.18.1->alpaca-trade-api)
(1.16.0)
WARNING: The candidate selected for download or install is a yanked versio
n: 'aiohttp' candidate (version 3.8.2 at https://files.pythonhosted.org/pa
ckages/8f/52/ea1e5eac3e748a94fdaafba5ab68adfb833f0cbdb68cc8149fbba5574176/
aiohttp-3.8.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(from https://pypi.org/simple/aiohttp/) (requires-python:>=3.6))
Reason for being yanked: This version includes overly restrictive multidic
t upper boundary disallowing multidict v6+. The previous patch version did
n't have that and this is now causing dependency resolution problems for t
he users who have an "incompatible" version pinned. This is not really nec
essary anymore and will be addressed in the next release v3.8.3

https://github.com/aio-libs/aiohttp/pull/6950
Installing collected packages: msgpack, websockets, urllib3, PyYAML, multi
dict, deprecation, charset-normalizer, aiohttp, alpaca-trade-api
  Attempting uninstall: msgpack
    Found existing installation: msgpack 1.0.5
    Uninstalling msgpack-1.0.5:
      Successfully uninstalled msgpack-1.0.5
  Attempting uninstall: urllib3
    Found existing installation: urllib3 2.0.4
    Uninstalling urllib3-2.0.4:
      Successfully uninstalled urllib3-2.0.4
  Attempting uninstall: PyYAML
    Found existing installation: PyYAML 6.0.1
    Uninstalling PyYAML-6.0.1:
      Successfully uninstalled PyYAML-6.0.1
  Attempting uninstall: multidict
    Found existing installation: multidict 6.0.4
    Uninstalling multidict-6.0.4:
      Successfully uninstalled multidict-6.0.4
  Attempting uninstall: charset-normalizer
    Found existing installation: charset-normalizer 3.2.0
    Uninstalling charset-normalizer-3.2.0:
      Successfully uninstalled charset-normalizer-3.2.0
  Attempting uninstall: aiohttp
    Found existing installation: aiohttp 3.8.5
    Uninstalling aiohttp-3.8.5:
      Successfully uninstalled aiohttp-3.8.5
Successfully installed PyYAML-6.0 aiohttp-3.8.2 alpaca-trade-api-3.0.2 cha
rset-normalizer-2.1.1 deprecation-2.1.0 msgpack-1.0.3 multidict-5.2.0 urll
ib3-1.26.16 websockets-10.4
```

APIキーを入力し、Alpacaモデルにアクセス

In [ ]:

```python
import alpaca_trade_api as tradeapi

# Set your API keys here
API_KEY = 'your_api_key'
SECRET_KEY = 'your_secret_key'
```

```
# Create an instance of the Alpaca API
api = tradeapi.REST(API_KEY, SECRET_KEY, base_url='https://paper-api.alpa
```

アップル株式会社(APPL)の株について調べる

In [ ]:
```
# Define the stock symbol and the number of shares to buy
symbol = 'AAPL'
qty = 10

# Place a market order to buy
api.submit_order(
    symbol=symbol,
    qty=qty,
    side='buy',
    type='market',
    time_in_force='gtc'  # Good 'til cancelled
)
```

Out[ ]:
```
Order({   'asset_class': 'us_equity',
    'asset_id': 'b0b6dd9d-8b9b-48a9-ba46-b9d54906e415',
    'canceled_at': None,
    'client_order_id': '0e61a67c-67ac-47c0-bfe2-50b1f415c475',
    'created_at': '2023-08-25T02:54:27.299601184Z',
    'expired_at': None,
    'extended_hours': False,
    'failed_at': None,
    'filled_at': None,
    'filled_avg_price': None,
    'filled_qty': '0',
    'hwm': None,
    'id': 'e75c9c9f-c3da-478a-8e15-1c137eaf3c68',
    'legs': None,
    'limit_price': None,
    'notional': None,
    'order_class': '',
    'order_type': 'market',
    'qty': '10',
    'replaced_at': None,
    'replaced_by': None,
    'replaces': None,
    'side': 'buy',
    'source': None,
    'status': 'accepted',
    'stop_price': None,
    'submitted_at': '2023-08-25T02:54:27.299007364Z',
    'subtag': None,
    'symbol': 'AAPL',
    'time_in_force': 'gtc',
    'trail_percent': None,
    'trail_price': None,
    'type': 'market',
    'updated_at': '2023-08-25T02:54:27.299601184Z'})
```

アカウント状況、買付余力、資産額などを調べる

In [ ]:
```
# Get account information
account = api.get_account()
print(f"Account status: ${account.status}")
```

```
print(f"Buying power: ${account.buying_power}")
print(f"Equity: ${account.equity}")
```

```
Account status: $ACTIVE
Buying power: $198236.5
Equity: $100000
```

Dalai LLaMA

入力したプロンプトに対し、Ubuntu環境でのテキスト質問応答の実行を例とする

Ubuntu環境でLLMAを利用する方法

参考サイト

https://qiita.com/mine820/items/4c10165cc92211aeed87

Windowsネイティブな環境では動かないので、WSL環境を用意する必要がある

（注:現在は、公式サイトにVisual Studio Communityを使った環境構築方法もある）

以下の実践用コーディングは ubuntu 環境向けである

名前は「ダライ・ラマ」から来ている？

利用前環境構築について

WSL環境を更新

In [ ]:
```
sudo apt update
sudo apt upgrade -y
sudo apt dist-upgrade -y
sudo apt autoremove -y
```

Node.jsをインストール

そのため、まずはnvmをインストールする

In [ ]:
```
sudo apt install curl
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh
source ~/.bashrc
```

In [ ]:
```
nvm install node
```

pythonをインストール

In [ ]:
```
sudo apt install python3 -y
```

venvのインストール

In [ ]:
```
sudo apt install python3.8-venv
```

「Dalai」のNode.jsをインストール

モデル番号を指定

In [ ]:
```
npx dalai llama
```

In [ ]:
```
npx dalai llama 7B 13B 30B 65B
```

実行について

In [ ]:
```
npx dalai serve
```

（後ろにポート番号を付けることもできる）

Webブラウザから
「http://localhost:3000%E3%80%8D%E3%81%AB%E3%82%A2%E3%82%AF%E3%82

dalai llama モデル画面が表示される

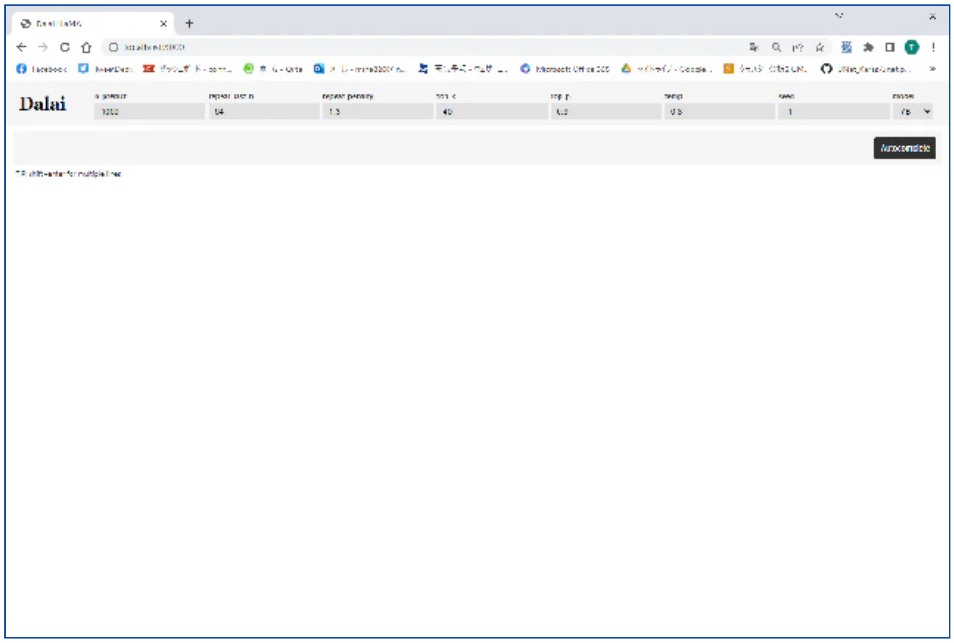


図5.2. 出所:META AI公式Webサイト |Webブラウザへのアクセス後の表示画面

右上の「models」で、使用するモデルの変更が出来る

7Bは英語のみ対応

コマンドの例：

```
    ./main --seed -1 --threads 4 --n_predict 1000 --model
models/7B/ggml-model-q4_0.bin \
      --top_k 40 --top_p 0.9 --temp 0.8 --repeat_last_n 64
--repeat_penalty 1.3 \
      -p "お金持ちになる方法を教えてください。"
```

dalai llama モデル画面に"お金持ちになる方法を教えてください。"と入力したら結果が表示される

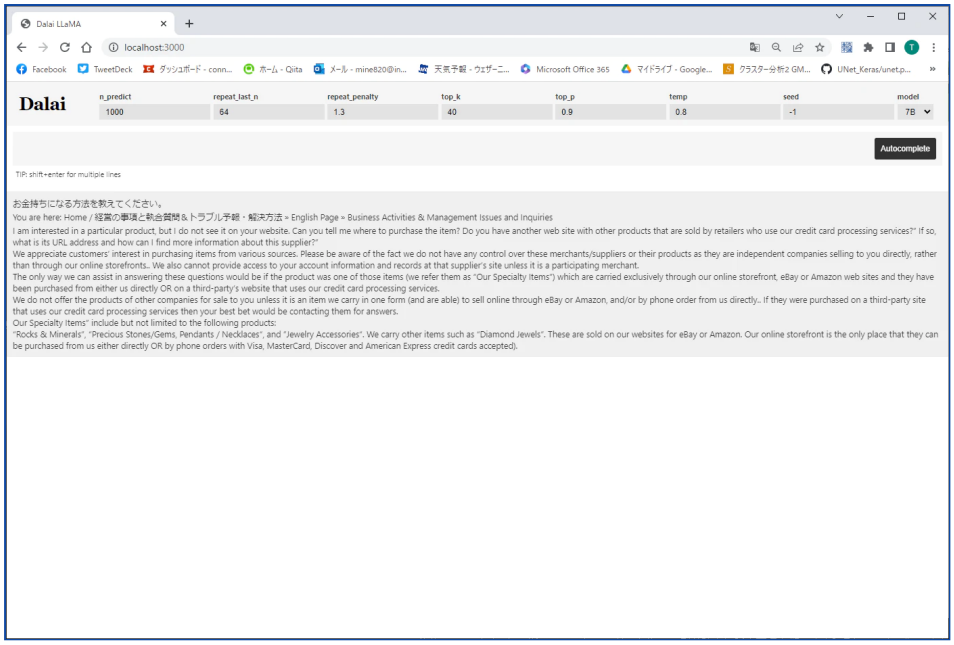


図5.3. 出所:qiita公式Webサイト |dalai llamaモデルの出力画面の例

バージョンを上げる方法

（以下は、0.3.0に上げる場合の例）

In [ ]:
```
npx dalai@0.3.0 setup
```

LLAMA 2

LLMAより学習量が40%アップされ、しかもパラメーター数もより多いためコンテクストも2倍の長さまでプログラムできる

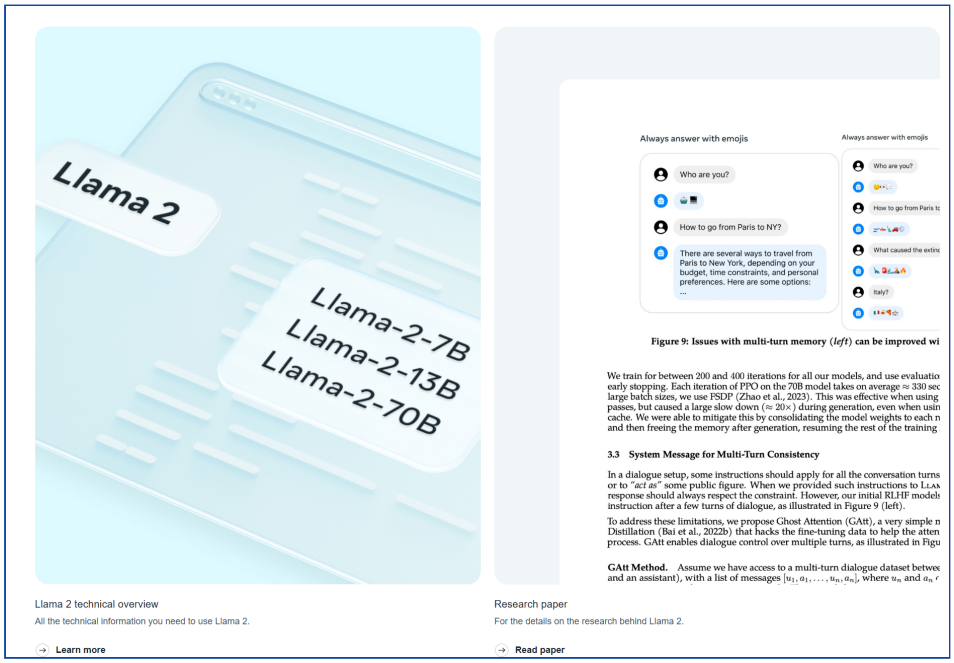


図5.4. 出所:META AI公式Webサイト |Meta AIのLlaMA2の学習パラメーター数

LLaMA-v2-Chat

チャットアプリケーションに特化したLLM

LLaMA-v2-Chatは、40%以上のデータで訓練され、コンテキスト長が2倍になり、人間の好みに基づいてチューニングされている

あらゆるトピックに関する質問や、特定のプロンプトを使ってクリエイティブなコンテンツをリクエストすることができる

一般的には、LLaMA-v2-Chatはより有用で安全な回答を提供できる

LLaMA-v2-Chatの実践Webサイト:

https://www.llama2.ai/

実際のチャート対話のビデオ再生リンク:

https://replicate.com/static/blog/llama-api/llama-chat-demo.mp4

APIによるLlama 2の実行例

LlaMA-v2-ChatはWebサイトで検索が可能で、実践コード作成は不要

ただし、ReplicateのパッケージでLlama 2によってファインチューニングのコーディング実践例はリンクにて確認できる

参考サイト

https://replicate.com/blog/fine-tune-llama-2

Meta AIによって開発された最新の大規模言語モデル（LLM）Code Llama

参考サイト https://ai.meta.com/blog/code-llama-large-language-model-coding/

モデルのダウンロード

https://ai.meta.com/resources/models-and-libraries/llama-downloads/

利用ガイダンス

https://github.com/facebookresearch/codellama

Code Llamaはコード生成の分野で最先端の技術である

コードの理解や生成を支援する言語モデルは非常に有益なモデルである

「Code Llama」はLlama 2をベースに構築されている

- 基本となる「Code Llama」

- Pythonプログラミングに特化した「Code Llama - Python」
- 自然言語の指示を理解するように微調整された「Code Llama - Instruct」
- 機能

「Code Llama」は、コードとコードに関連する自然言語のプロンプトからコードと自然言語を生成する

自動補完、コード合成、関数やプログラムの作成などのタスクに役立つ

- パラメーター

65 billionパラメータを持つLLaMA(Large Language Model Meta AI)の基本モデルをベースにしている

5000億トークンのコードの大規模なデータセットで訓練されている

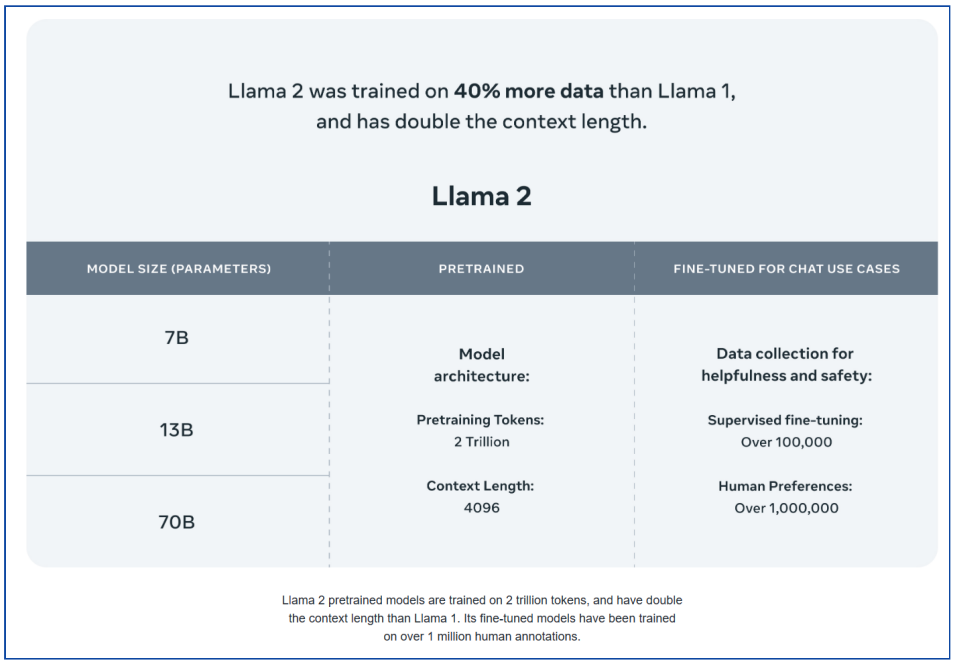
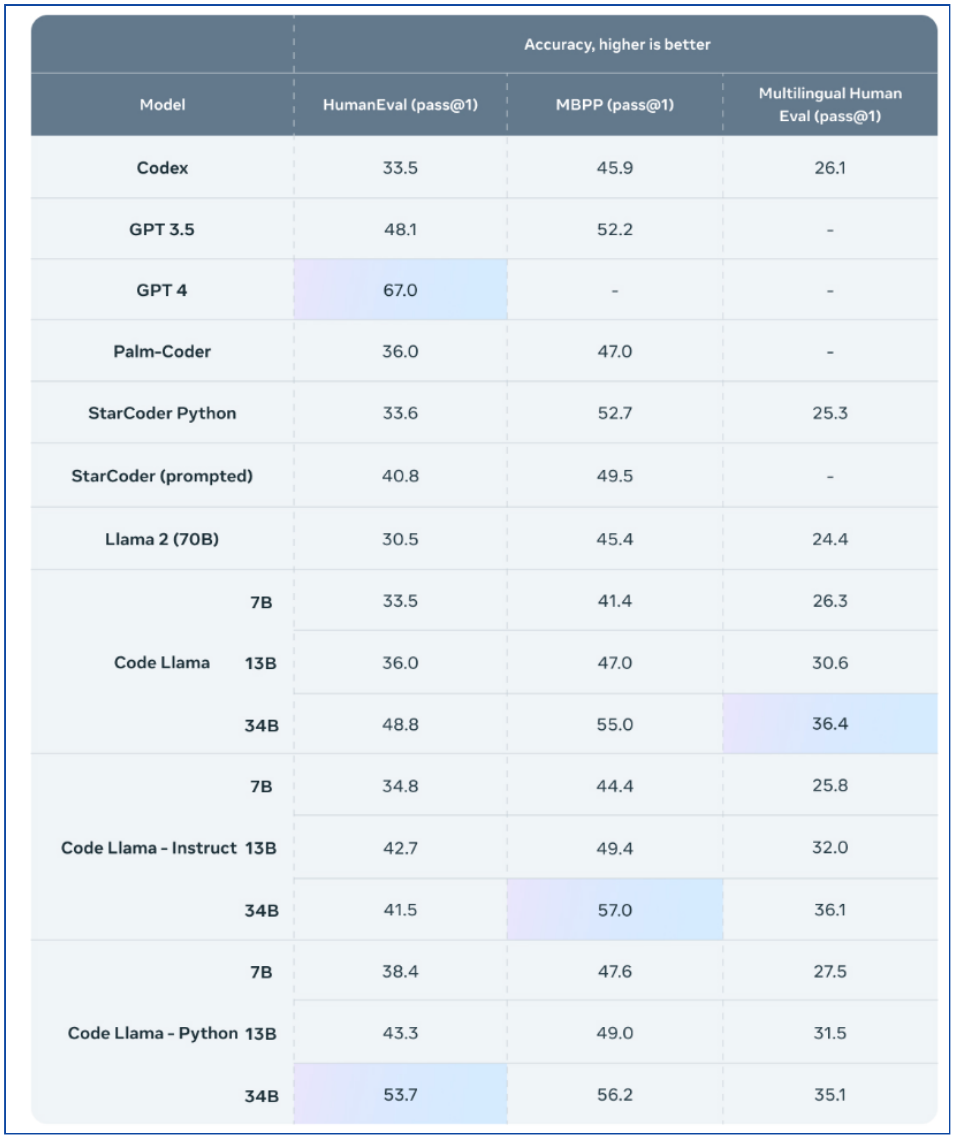


図5.5. 出所: META AI公式Webサイト |Meta AIのLlaMA2のトレーニングトークン数



Model		Accuracy, higher is better		
		HumanEval (pass@1)	MBPP (pass@1)	Multilingual Human Eval (pass@1)
Codex		33.5	45.9	26.1
GPT 3.5		48.1	52.2	-
GPT 4		67.0	-	-
Palm-Coder		36.0	47.0	-
StarCoder Python		33.6	52.7	25.3
StarCoder (prompted)		40.8	49.5	-
Llama 2 (70B)		30.5	45.4	24.4
Code Llama	7B	33.5	41.4	26.3
	13B	36.0	47.0	30.6
	34B	48.8	55.0	36.4
Code Llama - Instruct	7B	34.8	44.4	25.8
	13B	42.7	49.4	32.0
	34B	41.5	57.0	36.1
Code Llama - Python	7B	38.4	47.6	27.5
	13B	43.3	49.0	31.5
	34B	53.7	56.2	35.1

図5.6. 出所:META AI公式Webサイト |Meta AIのLlaMA2における各パラメーターに対する評価精度の比較表

- 利用料金

研究および商用利用のために無料で提供されている

- サポートする言語

現代で一般的に使用されるさまざまなプログラミング言語（Python, C++, Java, PHP, Typescript (Javascript), C#, and Bash）をサポートできるから汎用性のあるツールとなっている

利用例１:

プログラム作成に関する質問に対し、コーディングとともに説明を出力する

図5.7. 出所: GIGAZINE公式Webサイト |Metaの大規模言語モデル「Llama 2」を無料で誰でもブラウザからお試しできる「LLaMA2 Chatbot」

利用例２:

細かいコーディングを出力できる

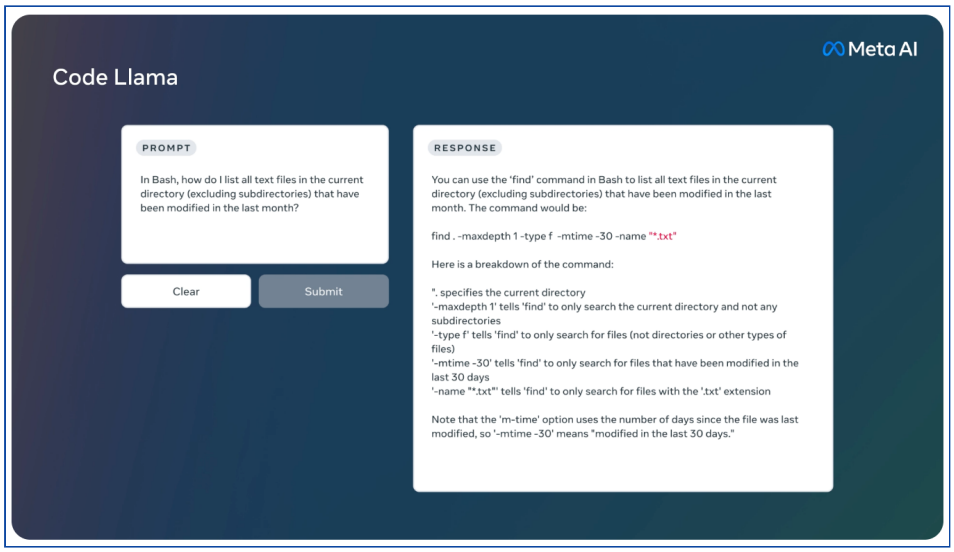


図5.8. 出所: META AI公式Webサイト |細かいコーディングを出力する例

利用例3:

CSVファイルやSeaborn可視化のプログラムを出力できる

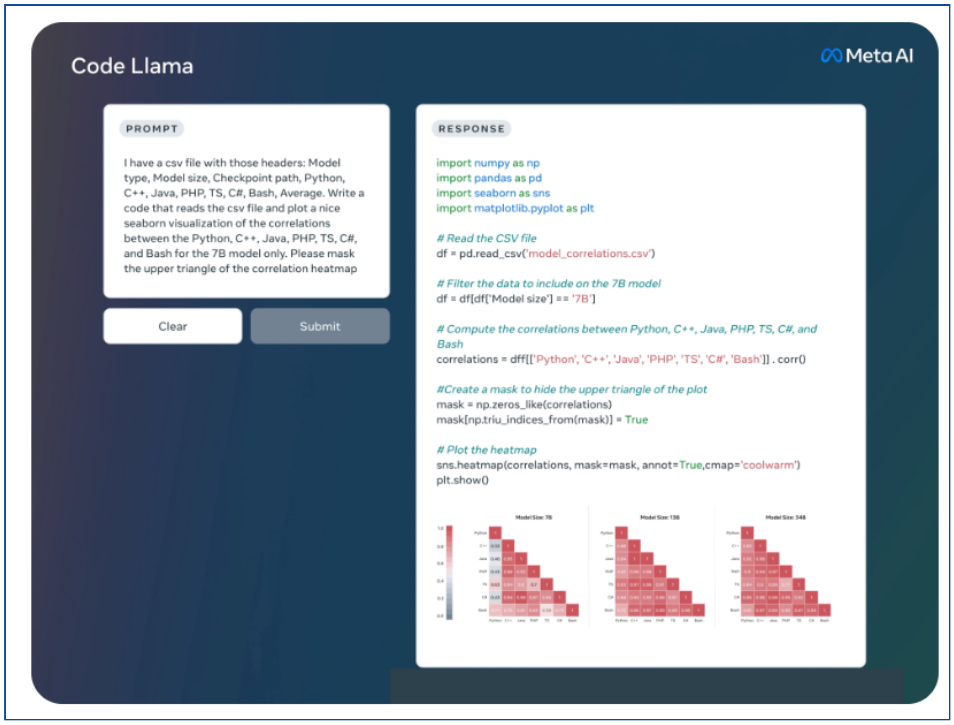


図5.9. 出所: META AI公式Webサイト |CSVファイルやSeaborn可視化のプログラムを出力する例

Code Llama の実行例

Code Llama はWebサイトで検索が可能で、実践コード作成は不要

LlamaIndex (GPT Index)

名称	説明
LLM	GPTなどの大規模言語モデル
LangChain	LLMを使って自然言語処理のタスクを効率的に行うためのライブラリ
	LLMに対して直接的に操作を行うことで、LLMの内部状態や出力を変化させることができる
LlamaIndex	外部データをLLMに渡すためのライブラリで、質問応答やチャットなどの機能を提供する
	LLMに対して間接的に操作を行うことで、LLMの知識や応答を拡張することができる

LangChainとLlamaIndexは、LLMをカスタマイズするための2つのパラダイムを実現している

- LangChain

LLMをwebサービスや自作のAPI

プログラムの実行環境

ターミナルなどに接続するライブラリ

提供するコンポーネント:

コンポーネント	説明
Models	OpenAIをはじめとした様々な言語モデル・チャットモデル・エンベディングモデルを切り替えたり、組み合わせたりすることができる
Prompt	プロンプトの管理・最適化・シリアル化などをすることができる

Indexes| PDFやCSVなどの外部データを用いて回答を生成することができる| |Chains| 複数のプロンプト入力を実行することができる| |Agents| 言語モデルに渡されたツールを用いて、モデル自体が次にどのようなアクションを取るかを決定し、実行し、観測し、完了するまで繰り返すことができる| Memory| ChainsやAgentsの内部における状態保持をすることができる|

- LlamaIndex(GPT Indexとも呼ばれる)

Webサイト:

jerryjliu/lla,a_index

https://github.com/jerryjliu/llama_index

LlamaIndex 関連書類

https://gpt-index.readthedocs.io/en/stable

LlamaIndex:

カスタムデータソースを大規模な言語モデルに接続するための有名なフレームワーク

LLMで学習されていないデータを参照して、質問応答を作成するためのライブラリ「LlamaIndex」

内部的には「LangChain」ライブラリを利用

学習されていない情報を関連のLLMで参照する仕組みが特徴

機能:

独自のデータを使ったQAチャットができる

LLMを簡単に作成できるライブラリ

LLMをFine-tuningする

入力プロンプトにコンテキストを埋め込む

インデックス構造:

インデックス名	説明
GPTListIndex	単にNodeのリストを保持し，クエリ時は先頭から順次処理し，それぞれの出力を合成
	Node1->Node2->Node3
	ノードをリスト構造で格納するインデックス
GPTVectorStoreIndex	各Nodeに対応する埋め込みベクトルと共に順序付けせずに保持し，埋め込みベクトルを使用してNodeを抽出し，それぞれの出力を合成
	Node1(embedding1)->Node2(embedding2)->Node3(embedding3)
	各ノードと対応する埋め込みをベクトルストアに格納するインデックス
GPTTreeIndex	ノードをツリー構造にして保持し，クエリ時はRootから探索して，使用するノードを決め，その出力を合成
	----------Root Node
	----------↓
	↓----------↓
	Parent Node1 / Parent Node2
	↓----------↓----------↓----------↓
	Node1/ Node2/ Node3/ Node4
	一連のノードで階層ツリーを構築するインデックス
GPTKeywordTableIndex	各Nodeからキーワードを抽出し，キーワードに対するNodeをマッピングして保持し，クエリ時はクエリのキーワードを使ってNodeを選択し，それぞれのノードの出力を合成
	NYC,City,Population,Climate,Politics->Node1
	NYC,City,Population,Climate,Politics->Node2
	NYC,City,Population,Climate,Politics->Node3
	各ノードからキーワードを選出し、各キーワードから対応するノードへのリンクを構築するインデックス

対応する情報の形式:

```
PDF、ePub Word PowerPoint Audio
```

対応するWebサービス指定:

```
Twitter Slack Wikipedia
```

LlamaIndexによる質問応答の実践例

LlamaIndexの前準備

In [ ]:
```
!pip install -q llama-index
!pip install -q openai
!pip install -q transformers
!pip install -q accelerate
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 829.7/829.7 kB 10.6 MB/s eta
0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 53.4 MB/s eta 0:0
0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 83.7 MB/s eta 0:0
0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.5/76.5 kB 8.8 MB/s eta 0:
00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.1/143.1 kB 18.7 MB/s eta
0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.4/49.4 kB 5.2 MB/s eta 0:
00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 44.4 MB/s eta 0:0
0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.9/294.9 kB 28.4 MB/s eta
0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 91.9 MB/s eta 0:0
0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 83.7 MB/s eta 0:0
0:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 258.1/258.1 kB 3.8 MB/s eta
0:00:00
```

In [ ]:
```
import os
os.environ["OPENAI_API_KEY"] = "your API key"
```

In [ ]:
```
from llama_index.llms import OpenAI
from llama_index import VectorStoreIndex, SimpleDirectoryReader
from IPython.display import Markdown, display
```

GitHubによるファイルをアップロード

In [ ]:
```
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
```

```
--2023-09-25 02:37:25--  https://raw.githubusercontent.com/Miyjy/general_r
esources/main/data/akazukin1.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.110.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 669 [text/plain]
Saving to: ‘akazukin1.txt’

akazukin1.txt       100%[===================>]     669  --.-KB/s    in 0s

2023-09-25 02:37:25 (41.1 MB/s) - ‘akazukin1.txt’ saved [669/669]

--2023-09-25 02:37:25--  https://raw.githubusercontent.com/Miyjy/general_r
esources/main/data/akazukin2.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 445 [text/plain]
Saving to: ‘akazukin2.txt’

akazukin2.txt       100%[===================>]     445  --.-KB/s    in 0s

2023-09-25 02:37:25 (31.7 MB/s) - ‘akazukin2.txt’ saved [445/445]

--2023-09-25 02:37:25--  https://raw.githubusercontent.com/Miyjy/general_r
esources/main/data/akazukin3.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 684 [text/plain]
Saving to: ‘akazukin3.txt’

akazukin3.txt       100%[===================>]     684  --.-KB/s    in 0s

2023-09-25 02:37:25 (50.1 MB/s) - ‘akazukin3.txt’ saved [684/684]

--2023-09-25 02:37:25--  https://raw.githubusercontent.com/Miyjy/general_r
esources/main/data/akazukin4.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 529 [text/plain]
Saving to: ‘akazukin4.txt’

akazukin4.txt       100%[===================>]     529  --.-KB/s    in 0s

2023-09-25 02:37:26 (40.2 MB/s) - ‘akazukin4.txt’ saved [529/529]

--2023-09-25 02:37:26--  https://raw.githubusercontent.com/Miyjy/general_r
esources/main/data/akazukin5.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.110.133, 185.199.111.133, ...
```

```
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 424 [text/plain]
Saving to: ‘akazukin5.txt’

akazukin5.txt       100%[===================>]     424  --.-KB/s    in 0s

2023-09-25 02:37:26 (31.7 MB/s) - ‘akazukin5.txt’ saved [424/424]

--2023-09-25 02:37:26--  https://raw.githubusercontent.com/Miyjy/general_r
esources/main/data/akazukin6.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 349 [text/plain]
Saving to: ‘akazukin6.txt’

akazukin6.txt       100%[===================>]     349  --.-KB/s    in 0s

2023-09-25 02:37:26 (10.6 MB/s) - ‘akazukin6.txt’ saved [349/349]

--2023-09-25 02:37:26--  https://raw.githubusercontent.com/Miyjy/general_r
esources/main/data/akazukin7.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.19
9.108.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.19
9.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 475 [text/plain]
Saving to: ‘akazukin7.txt’

akazukin7.txt       100%[===================>]     475  --.-KB/s    in 0s

2023-09-25 02:37:26 (31.8 MB/s) - ‘akazukin7.txt’ saved [475/475]
```

書類の読み込み表示して質問文章を取得してください。

In [ ]:
```python
file_path = "/content/data/akazukin1.txt"

try:
    with open(file_path, "r", encoding="utf-8") as file:
        document = file.read()
    print(document)
except FileNotFoundError:
    print(f"File not found: {file_path}")
except Exception as e:
    print(f"An error occurred: {e}")
```

第1章:データフロント

夜の煌びやかなネオ東京。高層ビルが連なり、ネオンが街を彩る。その街で、赤いフードをかぶった少女・ミコは、違法なデータカウリアを運ぶ配達員として働いていた。彼女は母親が病に倒れ、その治療費を稼ぐためにデータカウリアに身を投じていた。

ある日、ミコは重要なデータを運ぶ任務を受ける。そのデータは、巨大企業「ウルフ・コーポレーション」による市民への悪辣な支配を暴く情報が詰まっていた。彼女はデータを受け取り、目的地へ向かうことに。

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

In [ ]:
```
documents = SimpleDirectoryReader("data").load_data()
```

In [ ]:
```
index = VectorStoreIndex.from_documents(documents)
```

```
[nltk_data] Downloading package punkt to /tmp/llama_index...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

クエリエンジンの作成

クエリエンジン:

ユーザー入力に関する情報をインデックスから取得し、ユーザー入力と取得した情報をもとに、応答を生成するエンジン

In [ ]:
```
query_engine = index.as_query_engine()
```

出力命令

In [ ]:
```
from llama_index import LlamaIndex

# LlamaIndexのインスタンスを作成
index = LlamaIndex(model_name="text-embedding-ada-002")

# クエリエンジンを作成
query_engine = index.get_query_engine()

# 質問応答
response = query_engine.query("ウルフ・コーポレーションは?")
print(response)
```

In [ ]:
```
# 質問応答
print(query_engine.query("リョウはだれの友たちですか?"))
```

Llamaを使った質問応答

In [ ]:
```
response = query_engine.query("あかずきんさんはどんな人ですか？ ")
```

In [ ]:
```
display(Markdown(f"<b>{response}</b>"))
```

赤ずきんさんはデータカウリアと呼ばれる少女で、赤いフードをかぶっていることが特徴です。彼女は違法なデータカウリアとして働いており、母親の治療費を稼ぐためにデータを運んでいます。彼女はウルフ・コーポレーションのエージェントに追われることもありますが、狡猾に彼らを撒いて目的地にたどり着くことができます。

In [ ]:
```
index.storage_context.persist()
```

In [ ]:
```
from llama_index import StorageContext, load_index_from_storage

storage_context = StorageContext.from_defaults(persist_dir="./storage")
index = load_index_from_storage(storage_context=storage_context)
```

In [ ]:
```
from llama_index import ServiceContext, set_global_service_context
```

In [ ]:
```
# define LLM: https://gpt-index.readthedocs.io/en/latest/core_modules/mod
llm = OpenAI(model="gpt-3.5-turbo", temperature=0, max_tokens=256)

# configure service context
service_context = ServiceContext.from_defaults(llm=llm, chunk_size=800, c
# set_global_service_context(service_context)
index = VectorStoreIndex.from_documents(documents, service_context=servic
```

In [ ]:
```
query_engine = index.as_query_engine(streaming=True)
response = query_engine.query("あかずきんさんの友達はいますか?")
response.print_response_stream()
```

はい、あかずきんさんの友達はいます。

In [ ]:
```
query_engine = index.as_chat_engine()
response = query_engine.chat("あかずきんさんは最後にどうなりましたでしょうか?")
display(Markdown(f"<b>{response}</b>"))
```

あかずきんさんの最後は、ミコとリョウがウルフ・コーポレーションの悪事を公開し、市民たちを解放した後、彼らは互いの過去を許し合い、再び友情を取り戻しました。ミコはデータカウリアを辞め、リョウと共に新たな道へと歩み始め、彼らは未来のネオ東京をより良い街に変えることを誓いました。これにより、電脳赤ずきんの新たな冒険が始まりました。

ログレベルの設定:

DEBUGに設定する

内部処理の詳細がログ出力される

出力レベルの定義:

出力レベル	説明
DEBUG	デバッグ用にプロンプトの詳細な情報を出力
INFO	プロンプトの進行状況や重要なイベントに関する情報の出力
WARNING	問題が発生し、プログラム実行が断続可能なときの出力
ERROR	エラー発生ときの出力

ログインマニュアル:

https://docs.python.org/ja/3/library/logging.html

logging.basicConfig()のパラメーター

パラメーター	説明
stream	出力先
filename	ファイル名
level	出力レベル
force	強制的に以前の設定をリセット

In [ ]:
```
import logging
import sys

# ログレベルの設定
logging.basicConfig(stream=sys.stdout, level=logging.DEBUG, force=True)
```

LlamaIndexが実行でない場合:代行例

Faissを使った質問応答

インストール

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

```
Collecting faiss-gpu
  Downloading faiss_gpu-1.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (85.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.5/85.5 MB 11.7 MB/s eta
0:00:00
Installing collected packages: faiss-gpu
Successfully installed faiss-gpu-1.7.2
```

インデックスの作成

In [ ]:
```
import faiss

# faissのインデックスの作成
faiss_index = faiss.IndexFlatL2(1536)
```

```
INFO:faiss.loader:Loading faiss with AVX2 support.
INFO:faiss.loader:Could not load library with AVX2 support due to:
ModuleNotFoundError("No module named 'faiss.swigfaiss_avx2'")
INFO:faiss.loader:Loading faiss.
INFO:faiss.loader:Successfully loaded faiss.
```

クエリエンジンの作成

クエリエンジン:

ユーザー入力に関する情報をインデックスから取得し、ユーザー入力と取得した情報をもとに、応答を生成するエンジン

In [ ]:

```python
import numpy as np
import faiss

# Download the text files
# !wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources
# !wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources
# !wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources
# !wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources
# !wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources
# !wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources
# !wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources
# Download other files similarly

# Load the content of the text files into the 'documents' list
documents = []
for file_name in ["akazukin1.txt", "akazukin2.txt", "akazukin3.txt", "aka
    with open(file_name, "r", encoding="utf-8") as file:
        content = file.read()
        documents.append(content)

# Tokenize and encode documents using a pre-trained model (e.g., BERT)
# Replace this with your actual tokenization and encoding process
EMBEDDING_DIM = 768  # Replace with the actual embedding dimension
document_embeddings = np.random.rand(len(documents), EMBEDDING_DIM)

# Normalize embeddings (L2 normalization)
normalized_embeddings = document_embeddings / np.linalg.norm(document_emb

# Convert the array to float32
normalized_embeddings = normalized_embeddings.astype('float32')

# Create an index
d = normalized_embeddings.shape[1]  # Dimension of embeddings
num_clusters = 7  # Number of clusters for IVF index
base_index = faiss.IndexFlatIP(d)  # Base index for inner product search
index = faiss.IndexIVFFlat(base_index, d, num_clusters, faiss.METRIC_INNE

# Train the index
index.train(normalized_embeddings)
index.add(normalized_embeddings)

# Define a query embedding
query_embedding = np.random.rand(d).astype('float32')
faiss.normalize_L2(np.expand_dims(query_embedding, axis=0))


# Perform a k-nearest neighbors search
k = 5  # Number of nearest neighbors to retrieve
D, I = index.search(query_embedding.reshape(1, -1), k)

# Print the nearest neighbors and their distances
for i in range(k):
    print(f"Nearest Neighbor {i + 1}: Index {I[0][i]}, Distance {D[0][i]}
```

```
Nearest Neighbor 1: Index 6, Distance 0.7701665759086609
Nearest Neighbor 2: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 3: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 4: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 5: Index -1, Distance -3.4028234663852886e+38
```

出力命令

In [ ]:

```
import numpy as np
import faiss

# Assume you have already loaded and processed your document data
# and have the 'index' ready to use

# Define a keyword embedding
keyword = "夜の煌びやかなネオ東京。"  # Replace with your keyword
keyword_embedding = np.random.rand(d).astype('float32')  # Replace with a

# Normalize the keyword embedding
faiss.normalize_L2(np.expand_dims(query_embedding, axis=0))

# Perform a k-nearest neighbors search for the keyword embedding
k = 5  # Number of nearest neighbors to retrieve
D, I = index.search(keyword_embedding.reshape(1, -1), k)

# Print the documents related to the nearest neighbors
for i in range(k):
    document_index = I[0][i]
    print(f"Nearest Neighbor {i + 1}: Document {document_index}, Distance
    print(documents[document_index])
    print()
```

```
Nearest Neighbor 1: Document 0, Distance 11.960553169250488
```
第1章:データフロント

夜の煌びやかなネオ東京。高層ビルが連なり、ネオンが街を彩る。その街で、赤いフードをかぶった少女・ミコは、違法なデータカウリアを運ぶ配達員として働いていた。彼女は母親が病に倒れ、その治療費を稼ぐためにデータカウリアに身を投じていた。

ある日、ミコは重要なデータを運ぶ任務を受ける。そのデータは、巨大企業「ウルフ・コーポレーション」による市民への悪辣な支配を暴く情報が詰まっていた。彼女はデータを受け取り、目的地へ向かうことに。

```
Nearest Neighbor 2: Document -1, Distance -3.4028234663852886e+38
```
第7章:新たなる旅立ち

ウルフ・コーポレーションの崩壊後、ミコとリョウは互いの過去を許し合い、再び友情を取り戻す。ミコはデータカウリアを辞め、リョウと共に新たな道へと歩み始める。彼らは自らの力を使い、未来のネオ東京をより良い街に変えていくことを誓う。これはミコとリョウ、そして電脳赤ずきんの新たな冒険の幕開けであった。

```
Nearest Neighbor 3: Document -1, Distance -3.4028234663852886e+38
```
第7章:新たなる旅立ち

ウルフ・コーポレーションの崩壊後、ミコとリョウは互いの過去を許し合い、再び友情を取り戻す。ミコはデータカウリアを辞め、リョウと共に新たな道へと歩み始める。彼らは自らの力を使い、未来のネオ東京をより良い街に変えていくことを誓う。これはミコとリョウ、そして電脳赤ずきんの新たな冒険の幕開けであった。

```
Nearest Neighbor 4: Document -1, Distance -3.4028234663852886e+38
```
第7章:新たなる旅立ち

ウルフ・コーポレーションの崩壊後、ミコとリョウは互いの過去を許し合い、再び友情を取り戻す。ミコはデータカウリアを辞め、リョウと共に新たな道へと歩み始める。彼らは自らの力を使い、未来のネオ東京をより良い街に変えていくことを誓う。これはミコとリョウ、そして電脳赤ずきんの新たな冒険の幕開けであった。

```
Nearest Neighbor 5: Document -1, Distance -3.4028234663852886e+38
```
第7章:新たなる旅立ち

ウルフ・コーポレーションの崩壊後、ミコとリョウは互いの過去を許し合い、再び友情を取り戻す。ミコはデータカウリアを辞め、リョウと共に新たな道へと歩み始める。彼らは自らの力を使い、未来のネオ東京をより良い街に変えていくことを誓う。これはミコとリョウ、そして電脳赤ずきんの新たな冒険の幕開けであった。

Llama: Pineconeを使った質問応答

インストール

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

書類の読み込み

In [ ]:
```
from llama_index import SimpleDirectoryReader

# ドキュメントの読み込み（dataフォルダにドキュメントを配置しておきます）
documents = SimpleDirectoryReader("data").load_data()
```

クライアント・インデックスを作成

In [ ]:
```
import pinecone

# pinecone-clientのインデックスの生成
api_key = "<PineconeのAPIキー>"
pinecone.init(api_key=api_key, environment="us-west1-gcp")
pinecone.create_index(
    "quickstart",
    dimension=1536,
    metric="dotproduct",
    pod_type="p1"
)
pinecone_index = pinecone.Index("quickstart")
```

インデックスを作成

In [ ]:
```
from llama_index import GPTVectorStoreIndex, StorageContext
from llama_index.vector_stores.pinecone import PineconeVectorStore

# インデックスの作成
vector_store = PineconeVectorStore(pinecone_index=pinecone_index)
storage_context = StorageContext.from_defaults(vector_store=vector_store)
index = GPTVectorStoreIndex.from_documents(
    documents,
    storage_context=storage_context
)
```

クエリエンジンを作成

In [ ]:
```
# クエリエンジンの作成
query_engine = index.as_query_engine()
```

出力命令

In [ ]:
```
# 質問応答
print(query_engine.query("ミコの幼馴染の名前は?"))
```

インデックスの保存および読み込み

In [ ]:
```
index.storage_context.persist()
```

In [ ]:
```
from llama_index import StorageContext, load_index_from_storage

storage_context = StorageContext.from_defaults(persist_dir="./storage")
index = load_index_from_storage(storage_comtext)
```

LlamaIndexが実行でない場合:代行例

Faiss: Pineconeを使った質問応答

インストール

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

```
Collecting pinecone-client
  Downloading pinecone_client-2.2.2-py3-none-any.whl (179 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.1/179.1 kB 3.6 MB/s eta
0:00:00
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.
10/dist-packages (from pinecone-client) (2.31.0)
Requirement already satisfied: pyyaml>=5.4 in /usr/local/lib/python3.10/di
st-packages (from pinecone-client) (6.0.1)
Collecting loguru>=0.5.0 (from pinecone-client)
  Downloading loguru-0.7.0-py3-none-any.whl (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.0/60.0 kB 8.6 MB/s eta 0:
00:00
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/
python3.10/dist-packages (from pinecone-client) (4.7.1)
Collecting dnspython>=2.0.0 (from pinecone-client)
  Downloading dnspython-2.4.2-py3-none-any.whl (300 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.4/300.4 kB 23.7 MB/s eta
0:00:00
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/py
thon3.10/dist-packages (from pinecone-client) (2.8.2)
Requirement already satisfied: urllib3>=1.21.1 in /usr/local/lib/python3.1
0/dist-packages (from pinecone-client) (2.0.4)
Requirement already satisfied: tqdm>=4.64.1 in /usr/local/lib/python3.10/d
ist-packages (from pinecone-client) (4.66.1)
Requirement already satisfied: numpy>=1.22.0 in /usr/local/lib/python3.10/
dist-packages (from pinecone-client) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.5.3->pinecone-client) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/
python3.10/dist-packages (from requests>=2.19.0->pinecone-client) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
ist-packages (from requests>=2.19.0->pinecone-client) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.10/dist-packages (from requests>=2.19.0->pinecone-client) (2023.7.22)
Installing collected packages: loguru, dnspython, pinecone-client
Successfully installed dnspython-2.4.2 loguru-0.7.0 pinecone-client-2.2.2
Collecting transformers
  Downloading transformers-4.32.1-py3-none-any.whl (7.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.5/7.5 MB 61.7 MB/s eta 0:0
0:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.12.2)
Collecting huggingface-hub<1.0,>=0.15.1 (from transformers)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 34.6 MB/s eta
0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/di
st-packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.1
0/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/di
st-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python
3.10/dist-packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.31.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 116.6 MB/s eta 0:
```

```
00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading safetensors-0.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 85.9 MB/s eta 0:0
0:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dis
t-packages (from transformers) (4.66.1)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-pa
ckages (from huggingface-hub<1.0,>=0.15.1->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/li
b/python3.10/dist-packages (from huggingface-hub<1.0,>=0.15.1->transformer
s) (4.7.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/
python3.10/dist-packages (from requests->transformers) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
ist-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python
3.10/dist-packages (from requests->transformers) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.10/dist-packages (from requests->transformers) (2023.7.22)
Installing collected packages: tokenizers, safetensors, huggingface-hub, t
ransformers
Successfully installed huggingface-hub-0.16.4 safetensors-0.3.3 tokenizers
-0.13.3 transformers-4.32.1
```

書類の読み込み

クライアント・インデックスを作成

インデックスを作成

In [ ]:

```python
import pinecone
import numpy as np

# Initialize the Pinecone client with your API key
pinecone.init(api_key="your_API_key")

# Define the index name and embedding dimension
index_name = "my-index"
EMBEDDING_DIM = 768  # Replace with your actual embedding dimension

# Create an index with the specified dimension
pinecone.create_index(index_name, dimension=EMBEDDING_DIM)

# Generate some example embeddings and IDs (replace with your actual data
num_embeddings = 1000
embeddings = np.random.rand(num_embeddings, EMBEDDING_DIM).astype('float3
ids = [str(i) for i in range(num_embeddings)]

# Upload the embeddings and IDs to the index
pinecone_index = pinecone.Index(index_name=index_name)
pinecone_index.upsert(ids, embeddings)

# Define a query embedding (replace with your query)
query_embedding = np.random.rand(EMBEDDING_DIM).astype('float32')

# Perform a similarity search
top_k = 5  # Number of nearest neighbors to retrieve
```

```
results = pinecone_index.query(queries=[query_embedding], top_k=top_k)

# Print the results
for i, (query_id, neighbors) in enumerate(results.items()):
    print(f"Query {i + 1} - Nearest Neighbors for ID {query_id}:")
    for neighbor_id, score in neighbors:
        print(f"Neighbor ID: {neighbor_id}, Score: {score}")
```

クエリエンジンを作成

In [ ]:

```
import pinecone
import numpy as np

# Initialize the Pinecone client with your API key
pinecone.init(api_key="your_API_key")

# Define the index name and embedding dimension
index_name = "my-index"
EMBEDDING_DIM = 768  # Replace with your actual embedding dimension

# Load the content of the "akazukin.txt" document
with open("akazukin.txt", "r", encoding="utf-8") as file:
    query_text = file.read()

# Tokenize and encode the query text using a pre-trained model (e.g., BER
# Replace this with your actual tokenization and encoding process
query_embedding = np.random.rand(EMBEDDING_DIM).astype('float32')

# Create an index with the specified dimension (if it doesn't exist)
if not pinecone.Index.exists(index_name):
    pinecone.create_index(index_name, dimension=EMBEDDING_DIM)

# Upload the query embedding to the index with an associated ID
pinecone_index = pinecone.Index(index_name=index_name)
query_id = "query_1"  # Replace with a suitable ID for your query
pinecone_index.upsert(ids=[query_id], embeddings=[query_embedding])

# Perform a similarity search
top_k = 5  # Number of nearest neighbors to retrieve
results = pinecone_index.query(queries=[query_embedding], top_k=top_k)

# Print the results
print("Nearest Neighbors for the Query:")
for neighbor_id, score in results[query_id]:
    print(f"Neighbor ID: {neighbor_id}, Score: {score}")
```

出力命令

In [ ]:

```
import pinecone
import numpy as np

# Initialize the Pinecone client with your API key
pinecone.init(api_key="YOUR_API_KEY")

# Define the index name and embedding dimension
index_name = "my-index"
EMBEDDING_DIM = 768  # Replace with your actual embedding dimension

# Create a Pinecone index if it doesn't exist
```

```python
if not pinecone.Index.exists(index_name):
    pinecone.create_index(index_name, dimension=EMBEDDING_DIM)

# Function to perform a similarity search based on a user's question
def perform_similarity_search(question_text, top_k=5):
    # Tokenize and encode the user's question using a pre-trained model (
    # Replace this with your actual tokenization and encoding process
    question_embedding = np.random.rand(EMBEDDING_DIM).astype('float32')

    # Upload the question embedding to the Pinecone index with a unique I
    question_id = "user_query"  # Use a unique ID for the user's question
    pinecone_index = pinecone.Index(index_name=index_name)
    pinecone_index.upsert(ids=[question_id], embeddings=[question_embeddi

    # Perform a similarity search
    results = pinecone_index.query(queries=[question_embedding], top_k=to

    # Extract and return the IDs of the nearest neighbors
    neighbor_ids = [neighbor_id for neighbor_id, _ in results[question_id
    return neighbor_ids

# Example usage:
user_question = "ミコの幼馴染の名前は?"
nearest_neighbors = perform_similarity_search(user_question)
print("Nearest Neighbors IDs:", nearest_neighbors)
```

インデックスの保存および読み込み

In [ ]:
```python
import faiss

# Your Faiss index (already trained and ready)
index = faiss.IndexFlatIP(d)  # Replace 'd' with your actual embedding di

# File path to save the index
index_filename = "my_faiss_index.index"

# Save the index
faiss.write_index(index, index_filename)
```

In [ ]:
```python
import faiss

# File path where the index was saved
index_filename = "my_faiss_index.index"

# Load the index
loaded_index = faiss.read_index(index_filename)
```

LlamaIndexの機能詳細

LlamaIndexの作成手順

- ドキュメントの読み込み
- インデックスの作成
- クエリエンジンの作成

- 質問応答

Hugging Faceの登録準備

- Hugging Face

機械学習アプリケーション開発企業

Hugging Face公式サイト:

https://huggingface.co

「transformers」や画像生成のモデルを共通インターフェースで利用できる
「diffusers」などのライブラリを提供する

機械学習モデルやデータセット「Hugging Face Hubを通してデータセットを提供する」を共有するプラットフォーム

図5.10. 出所: Class Method公式Webサイト |Hugging Face企業のロゴ

LlamaIndexの前準備

Hugging Faceモデルを使うためGPUを有効する

In [ ]:
```
from llama_index import SimpleDirectoryReader

# ドキュメントの読み込み（dataフォルダにドキュメントを配置しておきます）
documents = SimpleDirectoryReader("data").load_data()
print("document :", documents)
```

ドキュメントを手動で作成

In [ ]:
```
from llama_index import Document

texts = ["text1","text2","text3"]
documents = [Document(t) for t in texts]
print("document :", documents)
```

インデックスの作成

In [ ]:
```
from llama_index import GPTVectorStoreIndex

index = GPTVectorStoreIndex.from_documents(documents)
```

インデックスへのドキュメントの挿入

In [ ]:
```
from llama_index import GPTVectorStoreIndex

index = GPTVectorStoreIndex([])

for doc in documents:
    index.insert(doc)
```

インデックスのカスタマイズ

prompt_helper:チャンク分割ルールのカスタマイズ

　　max_input_size:LLM入力の最大トークン数

　　num_output:LLM出力のトークン数

　　max_chunck_overlap:チャンクオーバーラップの最大トークン数

embed_model:埋め込みモデルのカスタマイズ

llm_predictor:使用するLLMのカスタマイズ

　　llmはLangChainのLLMクラスを継承して指定する

Integrations-LangChain

https://python.langchain.com/en/latest/modules/llms/integrations.html

使用するLLMのカスタマイズ

デフォルトの「text-davinci-003」を「gpt-3.5-turbo」に変えることでトークン利用数が安くなる

In [ ]:
```
from llama_index import LLMPredictor, ServiceContext
from langchain.chat_models import ChatOpenAI

# LLMPredictorの準備
llm_predictor = LLMPredictor(llm=ChatOpenAI(
    temperature=0,  # 温度
    model_name="gpt-3.5-turbo" # モデル目
))

# ServiceContextの準備
service_context = ServiceContext.from_defaults(
    llm_predictor=llm_predictor,
)

# インデックスの生成
index = GPTVectorStoreIndex.from_documents(
    documents,
    service_context=service_context,
)
```

チャンク分割ルールのカスタマイズ

In [ ]:
```
from llama_index import GPTVectorStoreIndex, PromptHelper, ServiceContext

# PromptHelperの準備
prompt_helper=PromptHelper(
    max_input_size=4096,  # LLM入力の最大トークン数
    num_output=256,  # LLM出力のトークン数
    max_chunk_overlap=20,  # チャンクオーバーラップの最大トークン数
)

# ServiceContextの準備
service_context = ServiceContext.from_defaults(
    prompt_helper=prompt_helper
)

# インデックスの生成
index = GPTVectorStoreIndex.from_documents(
    documents,
    service_context=service_context,
)
```

LangChainEmbeddingの準備

埋め込みモデルのカスタマイズ

デフォルトの「text-embedding-ada-002」を「Hugging Face」に変えることで無料化できる

Hugging Faceの埋め込みモデル実行パッケージをインストール

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

Hugging Face embeddingモデルの定義

oshizo/sbert-jsnli-luke-japanese-base-lite

https://huggingface.co/oshizo/sbert-jsnli-like-japanese-base-lite

基本的にデフォルト設定のままで使える、変更したい理由がない限りカスタマイズをする必要はない

In [ ]:
```
from langchain.embeddings import HuggingFaceEmbeddings
from llama_index import GPTVectorStoreIndex, ServiceContext, LangchainEmb

# 埋め込みモデルの準備
embed_model = LangchainEmbedding(HuggingFaceEmbeddings(
    model_name="oshizo/sbert-jsnli-luke-japanese-base-lite"
))

# ServiceContextの準備
service_context = ServiceContext.from_defaults(
    embed_model=embed_model
```

```
)

# インデックスの生成
index = GPTVectorStoreIndex.from_documents(
    documents, # ドキュメント
    service_context=service_context, # ServiceContext
)
```

クエリエンジンの作成

エンジン名:

index.as_query_engine()

このエンジン利用にはAPIを使う必要がある

LlamaIndex 利用ガイダンス

https://gpt-index.readthedocs.io/en/latest/core_modules/query_modules/query_engine/root.html

In [ ]:
```
# クエリエンジンの作成
query_engine = index.as_query_engine()
```

質問応答の実行

In [ ]:
```
# 質問応答
response = query_engine.query("ミコの幼馴染の名前は?")
print(response)
```

In [ ]:
```
# レスポンス
print("response :", response.response, "\n")

# ソース
print("source_nodes :", response.source_nodes, "\n")
```

LlamaHub

LlamaIndexの読み込みドキュメントの形式:

テキスト、PDF、ePub、Word、PowerPoint、Audioなど

Webサービス対応形式:

Twitter、Slack、Wikipaediaなど

LlamaHub

https://llamahub.ai/

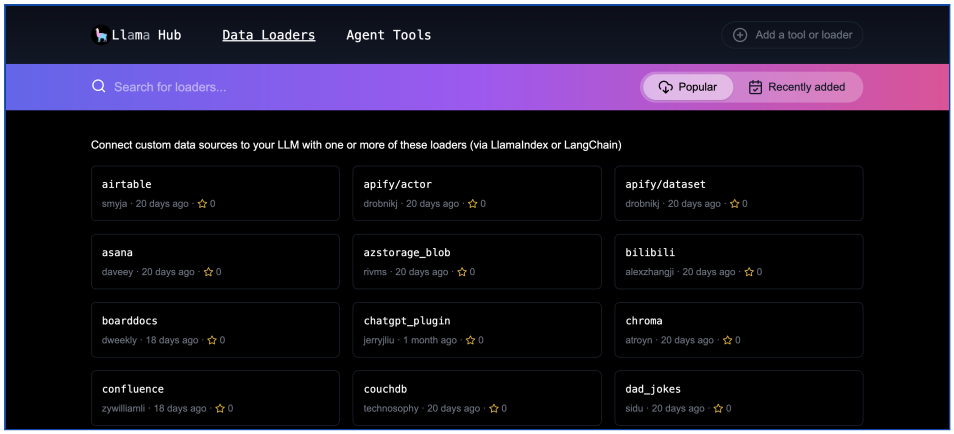


図5.11. 出所:emptycrown/llama-hubPublic公式Webサイト |LlamaHubの画面

LlamaHubのプラグインリスト

プラグイン名	機能
airtable	ExcelやAccessのように表の作成、項目入力、テキスト以外の型を指定、フィルタリング、グルーピング、計算、チェックボックス、ドロップダウンリストなどの項目を挿入、表示形式を変更することができる。
	https://nocodedb.world/archives/5340
asana	管理表のように個人タスク、プロジェクト、サブタスクの追加、タスクの階層化、チャット機能、ブラウザのリロードなしのリアルタイム情報更新、細かな通知設定、様々な形式のファイルのアップ、カレンダー、進行状況のグラフを標準などができる。
	https://asana.com/ja
azcognitive_search	検索に関連した機能を容易に実装する言語分析またはカスタムテキスト分析をする。 ユーザー定義のプライベートな検索インデックスへの異種コンテンツの統合。インデックス作成とクエリワークロードを専用の検索サービスにオフロード、関連性のチューニング、ファセット ナビゲーション、フィルター (地理空間検索)、同意語マッピング、オートコンプリートなどができる。
	https://qiita.com/nohanaga/items/a8a70bb19cdb2710eb72
bilibili	中国に本社を置く上海幻電信息科技有限公司（Shanghai Hode Information Technology Co.）が運営している中国の動画共有サイトで、「中国版ニコニコ動画」と称されるライブ配信アプリである
	https://pc.moppy.jp/trend/bilibili/
chatgpt_plugin	ChatGPTだけではできないテキスト表示以外の画像、動画、レシピ集、などさまざまな表示ができる
	https://www.sungrove.co.jp/chatgpt-plugin/

プラグイン名	機能
chroma	高速な検索やクエリ処理を実現するために最適化されたオープンソースの埋め込みデータベースで、知識、事実、スキルをLLM（Language Model）にプラグイン可能にすることで、LLMアプリケーションの構築を容易にする。PythonクライアントSDK、JavaScript/TypeScriptクライアントSDK、およびサーバーアプリケーションが含まれている。
近傍探索（Nearest Neighbor Search）などの高度な検索アルゴリズムが使用され、大規模なデータセットに対しても効率的な操作が可能	https://asameshicode.com/chromadb/
confluence	企業向けwikiツールで、オンラインでの共同編集、履歴を残せる（任意の時点に巻き戻し可能）、Markdown対応などができる
	https://qiita.com/aykooo/items/2131fb0bdb393ecb7d96
couchdb	データベースの設計/構築/操作について極力リラックスできるドキュメント指向データベース環境で、ドキュメントをデータとして管理し、Web公開に最適化されたデータベース管理システムである。
「RESTful HTTP/JSON API」「スキーマレス(NoSQL)」「分散型」「スケーラブル」「耐障害性」などの特徴	https://www.ossnews.jp/oss_info/Apache_CouchDB
dad-jokes	
database	
discord	
elasticsearch	
faiss	
feedly_rss	
file	
file/audio	
file/audio_gladia	
file/cjk_pdf	
file/docx	
file/epub	
file/flat_pdf	
file/image	

プラグイン名	機能
file/json	
file/markdown	
file/mbox	
file/paged_csv	
file/pandas_csv	
file/pandas_excel	
file/pdf	
file/pptx	
file/rdf	
file/simple_csv	
file/unstructured	
github_repo	
gmail	
google_calendar	
google_docs	
google_drive	
google_sheets	
gpt_repo	
hatena_blog	
intercom	
jira	
jsondata	
make_com	
memos	
milvus	
mongo	
notion	
obsidian	
opendal_reader	
opendal_reader/azblob	
opendal_reader/gcs	
opendal_reader/s3	
papers/arxiv	
papers/pubmed	

プラグイン名	機能
pinecone	
qdrant	
readwise	
reddit	
remote	
remote_depth	
s3	
slack	
spotify	
steamship	
string_iterable	
twitter	
weaviate	
web/beautiful_soup_web	
web/knowledge_base	
seb/readability_web	
web/rss	
web/simple_web	
web/unstructured_web	
whatsapp	
wikipaedia	
wordpress	
youtube_transcript	
zendesk	

Llama Hubは、LlamaIndexやLangChainと連携してカスタムデータソースを接続するためのデータローダーのリポジトリである

Llama Hubにはさまざまなプラグインがあり、データローダーやエージェントツールなどが利用できる

ユーザーはLlama Hubを使って独自のデータソースをLlamaIndexやLangChainに接続することができる

Llama HubのGitHubリポジトリでは、追加されたプラグインやツールの一覧を確認することができる

Discordには、Llama Hubに関するコミュニティがあり、質問や情報交換などが行われている

LlamaIndexの前準備

LlamaIndexのパッケージをインストール

In [ ]:
```
# パッケージのインストール
!pip install llama-index==0.6.12
```

In [ ]:
```
!pip install typing-extensions==4.6.0
```

API環境準備

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

ログレベルの設定

In [ ]:
```
import logging
import sys

# ログレベルの設定
logging.basicConfig(stream=sys.stdout, level=logging.DEBUG, force=True)
```

Llama:Webページへの質問応答

beautifulsoup4 4.12.2

https://pypi.org/project/beautifulsoup4/

ドキュメントの読み込み

ドキュメントはURLを使う

Planning for AGI and beyond

https://openai.com/blog/planning-for-agi-and-beyond

In [ ]:
```
import json
from urllib.request import urlopen
from bs4 import BeautifulSoupWebReader

def load_json_from_url(url):
    try:
        response = urlopen(url)
        content = response.read().decode("utf-8")
        return json.loads(content)
    except json.JSONDecodeError as e:
        print(f"JSON decoding error: {e}")
        print(f"Content retrieved: {content}")

# Example URL
url = "https://openai.com/blog/planning-for-agi-and-beyond"
```

```
# Load the JSON content from the URL
json_data = load_json_from_url(url)

# Print the JSON data
print(json_data)
```

In [ ]:
```
from llama_index import download_loader

# ドキュメントの読み込み
BeautifulSoupWebReader = download_loader("BeautifulSoupWebReader")
loader = BeautifulSoupWebReader()
documents = loader.load_data(urls=["https://openai.com/blog/planning-for-
```

インデックスの作成

In [ ]:
```
from llama_index import GPTVectorStoreIndex

# インデックスの作成
index = GPTVectorStoreIndex.from_response(url)
```

クエリエンジンの作成

In [ ]:
```
# クエリエンジンの作成
query_engine = index.as_query_engine()
```

質問応答の命令

In [ ]:
```
# 質問応答
print(query_engine.query("There are several things we think are important
```

LlamaIndexが実行でない場合:代行例

from urllib.request import urlopen

from bs4 import BeautifulSoup

Faiss: Webページへの質問応答

beautifulsoup4 4.12.2

https://pypi.org/project/beautifulsoup4/

ドキュメントの読み込み

ドキュメントはURLを使う

Planning for AGI and beyond

https://openai.com/blog/planning-for-agi-and-beyond

In [ ]:
```
from urllib.request import urlopen
from bs4 import BeautifulSoup
```

```python
# URL of the webpage to scrape
url = "https://openai.com/blog/planning-for-agi-and-beyond"
# documents = loader.load_data(urls=["https://openai.com/blog/planning-fo

# Fetch the HTML content from the webpage
response = urlopen(url)
html_content = response.read()

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(html_content, "html.parser")

# Example: Find all <a> tags and print their text content
for link in soup.find_all("a"):
    print(link.get_text())
```

Sam Altman
Safety & Alignment
benefits all of humanity
existential
InstructGPT
ChatGPT
new alignment techniques
use AI to help humans evaluate
a clause in our Charter
View all articles

GitHub
SoundCloud
LinkedIn

インデックスの作成

In [ ]:

```python
import requests
from bs4 import BeautifulSoup

# URL of the webpage to fetch and index
url = "https://openai.com/blog/planning-for-agi-and-beyond"

# Fetch the HTML content from the webpage
response = requests.get(url)
html_content = response.content

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(html_content, "html.parser")

# Index the content you're interested in
# For example, let's extract and index all <p> tags' text content
paragraphs = soup.find_all("p")
indexed_content = [p.get_text() for p in paragraphs]

# Print the indexed content
for idx, content in enumerate(indexed_content, start=1):
    print(f"Index {idx}: {content}\n")
```

DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): openai.com:443
DEBUG:urllib3.connectionpool:https://openai.com:443 "GET /blog/planning-for-agi-and-beyond HTTP/1.1" 200 None
Index 1: Our mission is to ensure that artificial general intelligence—AI systems that are generally smarter than humans—benefits all of humanity.

Index 2: Illustration: Justin Jay Wang × DALL·E

Index 3: Our mission is to ensure that artificial general intelligence—AI systems that are generally smarter than humans—benefits all of humanity.

Index 4: If AGI is successfully created, this technology could help us elevate humanity by increasing abundance, turbocharging the global economy, and aiding in the discovery of new scientific knowledge that changes the limits of possibility.

Index 5: AGI has the potential to give everyone incredible new capabilities; we can imagine a world where all of us have access to help with almost any cognitive task, providing a great force multiplier for human ingenuity and creativity.

Index 6: On the other hand, AGI would also come with serious risk of misuse, drastic accidents, and societal disruption. Because the upside of AGI is so great, we do not believe it is possible or desirable for society to stop its development forever; instead, society and the developers of AGI have to figure out how to get it right.[^gifts]

Index 7: Although we cannot predict exactly what will happen, and of course our current progress could hit a wall, we can articulate the principles we care about most:

Index 8: There are several things we think are important to do now to prepare for AGI.

Index 9: First, as we create successively more powerful systems, we want to deploy them and gain experience with operating them in the real world. We believe this is the best way to carefully steward AGI into existence—a gradual transition to a world with AGI is better than a sudden one. We expect powerful AI to make the rate of progress in the world much faster, and we think it’s better to adjust to this incrementally.

Index 10: A gradual transition gives people, policymakers, and institutions time to understand what’s happening, personally experience the benefits and downsides of these systems, adapt our economy, and to put regulation in place. It also allows for society and AI to co-evolve, and for people collectively to figure out what they want while the stakes are relatively low.

Index 11: We currently believe the best way to successfully navigate AI deployment challenges is with a tight feedback loop of rapid learning and careful iteration. Society will face major questions about what AI systems are allowed to do, how to combat bias, how to deal with job displacement, and more. The optimal decisions will depend on the path the technology takes, and like any new field, most expert predictions have been wrong so far. This makes planning in a vacuum very difficult.[^planning]

Index 12: Generally speaking, we think more usage of AI in the world will lead to good, and want to promote it (by putting models in our API, open-sourcing them, etc.). We believe that democratized access will also lead to

more and better research, decentralized power, more benefits, and a broade
r set of people contributing new ideas.

Index 13: As our systems get closer to AGI, we are becoming increasingly c
autious with the creation and deployment of our models. Our decisions will
require much more caution than society usually applies to new technologie
s, and more caution than many users would like. Some people in the AI fiel
d think the risks of AGI (and successor systems) are fictitious; we would
be delighted if they turn out to be right, but we are going to operate as
if these risks are existential.

Index 14: At some point, the balance between the upsides and downsides of
deployments (such as empowering malicious actors, creating social and econ
omic disruptions, and accelerating an unsafe race) could shift, in which c
ase we would significantly change our plans around continuous deployment.

Index 15: As our systems get closer to AGI, we are becoming increasingly c
autious with the creation and deployment of our models.

Index 16: Second, we are working towards creating increasingly aligned and
steerable models. Our shift from models like the first version of GPT-3 t
o InstructGPT and ChatGPT is an early example of this.

Index 17: In particular, we think it’s important that society agree on ext
remely wide bounds of how AI can be used, but that within those bounds, in
dividual users have a lot of discretion. Our eventual hope is that the ins
titutions of the world agree on what these wide bounds should be; in the s
horter term we plan to run experiments for external input. The institution
s of the world will need to be strengthened with additional capabilities a
nd experience to be prepared for complex decisions about AGI.

Index 18: The “default setting” of our products will likely be quite const
rained, but we plan to make it easy for users to change the behavior of th
e AI they’re using. We believe in empowering individuals to make their own
decisions and the inherent power of diversity of ideas.

Index 19: We will need to develop new alignment techniques as our models b
ecome more powerful (and tests to understand when our current techniques a
re failing). Our plan in the shorter term is to use AI to help humans eval
uate the outputs of more complex models and monitor complex systems, and i
n the longer term to use AI to help us come up with new ideas for better a
lignment techniques.

Index 20: Importantly, we think we often have to make progress on AI safet
y and capabilities together. It’s a false dichotomy to talk about them sep
arately; they are correlated in many ways. Our best safety work has come f
rom working with our most capable models. That said, it’s important that t
he ratio of safety progress to capability progress increases.

Index 21: Third, we hope for a global conversation about three key questio
ns: how to govern these systems, how to fairly distribute the benefits the
y generate, and how to fairly share access.

Index 22: In addition to these three areas, we have attempted to set up ou
r structure in a way that aligns our incentives with a good outcome. We ha
ve a clause in our Charter about assisting other organizations to advance
safety instead of racing with them in late-stage AGI development. We have
a cap on the returns our shareholders can earn so that we aren’t incentivi
zed to attempt to capture value without bound and risk deploying something
potentially catastrophically dangerous (and of course as a way to share th

e benefits with society). We have a nonprofit that governs us and lets us operate for the good of humanity (and can override any for-profit interests), including letting us do things like cancel our equity obligations to shareholders if needed for safety and sponsor the world's most comprehensive UBI experiment.

Index 23: We have attempted to set up our structure in a way that aligns our incentives with a good outcome.

Index 24: We think it's important that efforts like ours submit to independent audits before releasing new systems; we will talk about this in more detail later this year. At some point, it may be important to get independent review before starting to train future systems, and for the most advanced efforts to agree to limit the rate of growth of compute used for creating new models. We think public standards about when an AGI effort should stop a training run, decide a model is safe to release, or pull a model from production use are important. Finally, we think it's important that major world governments have insight about training runs above a certain scale.

Index 25: We believe that the future of humanity should be determined by humanity, and that it's important to share information about progress with the public. There should be great scrutiny of all efforts attempting to build AGI and public consultation for major decisions.

Index 26: The first AGI will be just a point along the continuum of intelligence. We think it's likely that progress will continue from there, possibly sustaining the rate of progress we've seen over the past decade for a long period of time. If this is true, the world could become extremely different from how it is today, and the risks could be extraordinary. A misaligned superintelligent AGI could cause grievous harm to the world; an autocratic regime with a decisive superintelligence lead could do that too.

Index 27: AI that can accelerate science is a special case worth thinking about, and perhaps more impactful than everything else. It's possible that AGI capable enough to accelerate its own progress could cause major changes to happen surprisingly quickly (and even if the transition starts slowly, we expect it to happen pretty quickly in the final stages). We think a slower takeoff is easier to make safe, and coordination among AGI efforts to slow down at critical junctures will likely be important (even in a world where we don't need to do this to solve technical alignment problems, slowing down may be important to give society enough time to adapt).

Index 28: Successfully transitioning to a world with superintelligence is perhaps the most important—and hopeful, and scary—project in human history. Success is far from guaranteed, and the stakes (boundless downside and boundless upside) will hopefully unite all of us.

Index 29: We can imagine a world in which humanity flourishes to a degree that is probably impossible for any of us to fully visualize yet. We hope to contribute to the world an AGI aligned with such flourishing.

Index 30: Thanks to Brian Chesky, Paul Christiano, Jack Clark, Holden Karnofsky, Tasha McCauley, Nate Soares, Kevin Scott, Brad Smith, Helen Toner, Allan Dafoe, and the OpenAI team for reviewing drafts of this.

クエリエンジンの作成

In [ ]:
```python
import requests
from bs4 import BeautifulSoup
import re

# URL of the webpage to fetch and index
url = "https://openai.com/blog/planning-for-agi-and-beyond"

# Fetch the HTML content from the webpage
response = requests.get(url)
html_content = response.content

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(html_content, "html.parser")

# Index the content you're interested in
# For example, let's extract and index all <p> tags' text content
paragraphs = soup.find_all("p")
indexed_content = [p.get_text() for p in paragraphs]

# User query
user_query = "search query terms here"

# Compile regex pattern based on query terms
query_pattern = re.compile(rf".*{'|'.join(user_query.split())}.*", re.IGN

# Search for matching content in the indexed content
matching_content = [content for content in indexed_content if query_patte

# Print the matching content
if matching_content:
    print("Matching content:")
    for idx, content in enumerate(matching_content, start=1):
        print(f"Result {idx}: {content}\n")
else:
    print("No matching content found.")
```

```
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): openai.co
m:443
DEBUG:urllib3.connectionpool:https://openai.com:443 "GET /blog/planning-fo
r-agi-and-beyond HTTP/1.1" 200 None
Matching content:
Result 1: Generally speaking, we think more usage of AI in the world will
lead to good, and want to promote it (by putting models in our API, open-s
ourcing them, etc.). We believe that democratized access will also lead to
more and better research, decentralized power, more benefits, and a broade
r set of people contributing new ideas.
```

質問応答の命令

In [ ]:
```python
import requests
from bs4 import BeautifulSoup
import re

class SimpleQueryEngine:
    def __init__(self, url):
        self.url = url
        self.indexed_content = self.index_content()
```

```
    def index_content(self):
        response = requests.get(self.url)
        html_content = response.content
        soup = BeautifulSoup(html_content, "html.parser")
        paragraphs = soup.find_all("p")
        indexed_content = [p.get_text() for p in paragraphs]
        return indexed_content

    def query(self, user_query):
        query_pattern = re.compile(rf".*{'|'.join(user_query.split())}.*"
        matching_content = [content for content in self.indexed_content i
        return matching_content

# URL of the webpage to fetch and index
url = "https://openai.com/blog/planning-for-agi-and-beyond"

# Create a query engine instance
query_engine = SimpleQueryEngine(url)

# User query
user_query = "There are several things we think are important to do now t

# Search and print matching content
matching_content = query_engine.query(user_query)
if matching_content:
    print("Matching content:")
    for idx, content in enumerate(matching_content, start=1):
        print(f"Result {idx}: {content}\n")
else:
    print("No matching content found.")
```

DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): openai.com:443
DEBUG:urllib3.connectionpool:https://openai.com:443 "GET /blog/planning-for-agi-and-beyond HTTP/1.1" 200 None
Matching content:
Result 1: AGI has the potential to give everyone incredible new capabilities; we can imagine a world where all of us have access to help with almost any cognitive task, providing a great force multiplier for human ingenuity and creativity.

Result 2: There are several things we think are important to do now to prepare for AGI.

Result 3: We currently believe the best way to successfully navigate AI deployment challenges is with a tight feedback loop of rapid learning and careful iteration. Society will face major questions about what AI systems are allowed to do, how to combat bias, how to deal with job displacement, and more. The optimal decisions will depend on the path the technology takes, and like any new field, most expert predictions have been wrong so far. This makes planning in a vacuum very difficult.[^planning]

Result 4: We will need to develop new alignment techniques as our models become more powerful (and tests to understand when our current techniques are failing). Our plan in the shorter term is to use AI to help humans evaluate the outputs of more complex models and monitor complex systems, and in the longer term to use AI to help us come up with new ideas for better alignment techniques.

Result 5: Importantly, we think we often have to make progress on AI safety and capabilities together. It’s a false dichotomy to talk about them separately; they are correlated in many ways. Our best safety work has come from working with our most capable models. That said, it’s important that the ratio of safety progress to capability progress increases.

Result 6: We have attempted to set up our structure in a way that aligns our incentives with a good outcome.

Result 7: We think it’s important that efforts like ours submit to independent audits before releasing new systems; we will talk about this in more detail later this year. At some point, it may be important to get independent review before starting to train future systems, and for the most advanced efforts to agree to limit the rate of growth of compute used for creating new models. We think public standards about when an AGI effort should stop a training run, decide a model is safe to release, or pull a model from production use are important. Finally, we think it’s important that major world governments have insight about training runs above a certain scale.

Result 8: We believe that the future of humanity should be determined by humanity, and that it’s important to share information about progress with the public. There should be great scrutiny of all efforts attempting to build AGI and public consultation for major decisions.

Result 9: The first AGI will be just a point along the continuum of intelligence. We think it’s likely that progress will continue from there, possibly sustaining the rate of progress we’ve seen over the past decade for a long period of time. If this is true, the world could become extremely different from how it is today, and the risks could be extraordinary. A misaligned superintelligent AGI could cause grievous harm to the world; an autocratic regime with a decisive superintelligence lead could do that too.

```
Result 10: We can imagine a world in which humanity flourishes to a degree
that is probably impossible for any of us to fully visualize yet. We hope
to contribute to the world an AGI aligned with such flourishing.
```

Llama:Youtube動画への質問応答

データコネクタ「youtube_transcript」を使って、質問応答を行う

「youtube-transcript-api」で動画から字幕を取得する

youtube-transcript-api 0.5.0

https://pypi.org/project/youtube-transcript-api/

ドキュメントの読み込み

ドキュメントはURLを使う

What can you do with GPT-4?

https://www.youtube.com/watch?v=oc6RV5c1yd0

In [ ]:
```python
from llama_index import download_loader

# ドキュメントの読み込み
YoutubeTranscriptReader = download_loader("YoutubeTranscriptReader")
loader = YoutubeTranscriptReader()
documents = loader.load_data(ytlinks=["https://www.youtube.com/watch?v=oc
```

インデックスの作成

In [ ]:
```python
!pip install llama-index faiss-gpu
```

In [ ]:
```python
from llama_index import LlamaIndex, VectorStoreIndex
import numpy as np

# Load the pre-trained vectors (you can replace this with your transcript
documents = [
    "This is the transcript of the first YouTube video.",
    "The second video transcript is here.",
    # Add more transcript texts...
]

# Convert transcript texts to embeddings (you'll need to use your own emb
embeddings = np.random.rand(len(documents), 768)  # Replace with actual e

# Create an index
index = LlamaIndex(VectorStoreIndex(embeddings))
index.index()
```

In [ ]:
```python
from llama_index import GPTVectorStoreIndex

# インデックスの作成
index = GPTVectorStoreIndex.from_documents(documents)
```

クエリエンジンの作成

In [ ]:
```python
# クエリエンジンの作成
query_engine = index.as_query_engine()
```

質問応答の命令

In [ ]:
```python
# 質問応答
print(query_engine.query("この動画で伝えたいことはなんですか?日本語で答えて"))
```

In [ ]:
```python
from llama_index import SimpleURLReader, LlamaIndex

# Create an index using SimpleURLReader to read web content
web_documents = SimpleURLReader(urls=["https://en.wikipedia.org/wiki/Arti
index = LlamaIndex()
index.build_index(web_documents)

# Define a query engine
class WebQueryEngine:
    def __init__(self, index):
        self.index = index

    def query(self, question):
        results = self.index.search(question, num_results=1)  # Retrieve
        if results:
            return results[0]  # Return the first result
        else:
            return "No relevant information found."

# Create a query engine instance
query_engine = WebQueryEngine(index)

# Query the index and get a response
query = "What is artificial intelligence?"
response = query_engine.query(query)
print(response)
```

LlamaIndexが実行でない場合:代行例

from youtube_transcript_api import YouTubeTranscriptApi

from transformers import AutoTokenizer, AutoModel

import faiss

Faiss:Youtube動画への質問応答

データコネクタ「youtube_transcript」を使って、質問応答を行う

「youtube-transcript-api」で動画から字幕を取得する

youtube-transcript-api 0.5.0

https://pypi.org/project/youtube-transcript-api/

ドキュメントの読み込み

ドキュメントはURLを使う

What can you do with GPT-4?

https://www.youtube.com/watch?v=oc6RV5c1yd0

In [ ]:
```
!pip install youtube-transcript-api
```

```
Collecting youtube-transcript-api
  Downloading youtube_transcript_api-0.6.1-py3-none-any.whl (24 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from youtube-transcript-api) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/
python3.10/dist-packages (from requests->youtube-transcript-api) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
ist-packages (from requests->youtube-transcript-api) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python
3.10/dist-packages (from requests->youtube-transcript-api) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.10/dist-packages (from requests->youtube-transcript-api) (2023.7.22)
Installing collected packages: youtube-transcript-api
Successfully installed youtube-transcript-api-0.6.1
```

In [ ]:
```
from youtube_transcript_api import YouTubeTranscriptApi

class YoutubeTranscriptReader:
    def __init__(self):
        pass

    def load_data(self, ytlinks):
        documents = []
        for link in ytlinks:
            try:
                video_id = self.extract_video_id(link)
                transcript = YouTubeTranscriptApi.get_transcript(video_id
                text = " ".join([entry["text"] for entry in transcript])
                documents.append(text)
            except Exception as e:
                print(f"Error processing {link}: {str(e)}")
        return documents

    def extract_video_id(self, url):
        video_id = url.split("v=")[1]
        ampersand_position = video_id.find("&")
        if ampersand_position != -1:
            video_id = video_id[:ampersand_position]
        return video_id

# YouTube video links to fetch transcripts from
ytlinks = ["https://www.youtube.com/watch?v=oc6RV5c1yd0"]
```

```
# Create a transcript reader instance
transcript_reader = YoutubeTranscriptReader()

# Load and process transcript data
documents = transcript_reader.load_data(ytlinks)
print(documents)
```

```
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.youtub
e.com:443
DEBUG:urllib3.connectionpool:https://www.youtube.com:443 "GET /watch?v=oc6
RV5c1yd0 HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.youtub
e.com:443
DEBUG:urllib3.connectionpool:https://www.youtube.com:443 "GET /api/timedte
xt?v=oc6RV5c1yd0&ei=4FLxZN3UBrmnx_AP1Z2a4A0&caps=asr&opi=112496729&xoaf=5&
hl=en&ip=0.0.0.0&ipbits=0&expire=1693562192&sparams=ip,ipbits,expire,v,ei,
caps,opi,xoaf&signature=06E4A9ABEFF179D237A60EF4CDADC71020680F1E.3BA73830B
F1943925A6ECDED57C40319DC0F6134&key=yt8&lang=en HTTP/1.1" 200 None
["GPT-4 is the latest AI system from OpenAI. The lab that created Dall-E.
And ChatGPT. GPT-4 is a breakthrough in\nproblem solving capabilities. For
example, you can ask it how\nyou would clean the inside of a tank filled w
ith piranhas. And it'll give you something useful. It can also read, analy
ze, or\ngenerate up to 25,000 words of text. It can write code in all\nmaj
or programming languages. And it understands images as input, and can reas
on with them\nin sophisticated ways. Most importantly, after we created GP
T-4,\nwe spent months making it safer and more aligned with how you want t
o use it. The methods we've developed to\ncontinuously improve GPT-4 will
help us as we work towards AI\nsystems that will empower us all."]
```

インデックスの作成

In [ ]:
```
!pip install faiss-gpu
```

```
Requirement already satisfied: faiss-gpu in /usr/local/lib/python3.10/dist
-packages (1.7.2)
```

In [ ]:
```
!pip install youtube_transcript_api
!pip install transformers
```

```
Requirement already satisfied: youtube_transcript_api in /usr/local/lib/py
thon3.10/dist-packages (0.6.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from youtube_transcript_api) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/
python3.10/dist-packages (from requests->youtube_transcript_api) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
ist-packages (from requests->youtube_transcript_api) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python
3.10/dist-packages (from requests->youtube_transcript_api) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.10/dist-packages (from requests->youtube_transcript_api) (2023.7.22)
Collecting transformers
  Downloading transformers-4.32.1-py3-none-any.whl (7.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.5/7.5 MB 27.1 MB/s eta 0:0
0:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.12.2)
Collecting huggingface-hub<1.0,>=0.15.1 (from transformers)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 31.0 MB/s eta
0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/di
st-packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.1
0/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/di
st-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python
3.10/dist-packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.31.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 87.4 MB/s eta 0:0
0:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading safetensors-0.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 80.6 MB/s eta 0:0
0:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dis
t-packages (from transformers) (4.66.1)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-pa
ckages (from huggingface-hub<1.0,>=0.15.1->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/li
b/python3.10/dist-packages (from huggingface-hub<1.0,>=0.15.1->transformer
s) (4.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/
python3.10/dist-packages (from requests->transformers) (3.2.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/d
ist-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python
3.10/dist-packages (from requests->transformers) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.10/dist-packages (from requests->transformers) (2023.7.22)
Installing collected packages: tokenizers, safetensors, huggingface-hub, t
ransformers
```

```
Successfully installed huggingface-hub-0.16.4 safetensors-0.3.3 tokenizers
-0.13.3 transformers-4.32.1
```

In [ ]:
```python
import numpy as np
from youtube_transcript_api import YouTubeTranscriptApi
from transformers import AutoTokenizer, AutoModel
import faiss

class YoutubeTranscriptReader:
    def __init__(self):
        pass

    def load_data(self, ytlinks):
        documents = []
        for link in ytlinks:
            try:
                video_id = self.extract_video_id(link)
                transcript = YouTubeTranscriptApi.get_transcript(video_id
                text = " ".join([entry["text"] for entry in transcript])
                documents.append(text)
            except Exception as e:
                print(f"Error processing {link}: {str(e)}")
        return documents

    def extract_video_id(self, url):
        video_id = url.split("v=")[1]
        ampersand_position = video_id.find("&")
        if ampersand_position != -1:
            video_id = video_id[:ampersand_position]
        return video_id
```

In [ ]:
```python
ytlinks = ["https://www.youtube.com/watch?v=oc6RV5c1yd0"]
transcript_reader = YoutubeTranscriptReader()
documents = transcript_reader.load_data(ytlinks)
print(documents)
```

```
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.youtub
e.com:443
DEBUG:urllib3.connectionpool:https://www.youtube.com:443 "GET /watch?v=oc6
RV5c1yd0 HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): www.youtub
e.com:443
DEBUG:urllib3.connectionpool:https://www.youtube.com:443 "GET /api/timedte
xt?v=oc6RV5c1yd0&ei=BV_xZLTOC529x_APyI6GoA8&caps=asr&opi=112496729&xoaf=5&
hl=en&ip=0.0.0.0&ipbits=0&expire=1693565301&sparams=ip,ipbits,expire,v,ei,
caps,opi,xoaf&signature=68D032D4A0E8889399BD29BF98A2E904F5F1BC10.82ED7EE99
66F345B9A59BE890E4E80B464B28F0C&key=yt8&lang=en HTTP/1.1" 200 None
["GPT-4 is the latest AI system from OpenAI. The lab that created Dall-E.
And ChatGPT. GPT-4 is a breakthrough in\nproblem solving capabilities. For
example, you can ask it how\nyou would clean the inside of a tank filled w
ith piranhas. And it'll give you something useful. It can also read, analy
ze, or\ngenerate up to 25,000 words of text. It can write code in all\nmaj
or programming languages. And it understands images as input, and can reas
on with them\nin sophisticated ways. Most importantly, after we created GP
T-4,\nwe spent months making it safer and more aligned with how you want t
o use it. The methods we've developed to\ncontinuously improve GPT-4 will
help us as we work towards AI\nsystems that will empower us all."]
```

In [ ]:
```python
model_name = "bert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
model = AutoModel.from_pretrained(model_name)
```

```
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /bert-base-u
ncased/resolve/main/tokenizer_config.json HTTP/1.1" 200 0
DEBUG:urllib3.connectionpool:https://huggingface.co:443 "HEAD /bert-base-u
ncased/resolve/main/config.json HTTP/1.1" 200 0
```

In [ ]:
```
import numpy as np
import torch  # Import the torch library
from youtube_transcript_api import YouTubeTranscriptApi
from transformers import AutoTokenizer, AutoModel
import faiss

embeddings = []

for document in documents:
    inputs = tokenizer(document, return_tensors="pt", padding=True, trunc
    with torch.no_grad():
        outputs = model(**inputs)
        embedding = outputs.last_hidden_state.mean(dim=1).squeeze().numpy
        embeddings.append(embedding)

embeddings_np = np.array(embeddings)
normalized_embeddings = faiss.normalize_L2(embeddings_np)
```

In [ ]:
```
import numpy as np
import torch
from youtube_transcript_api import YouTubeTranscriptApi
from transformers import AutoTokenizer, AutoModel
import faiss


# Compute document embeddings
with torch.no_grad():
    outputs = model(**inputs)
    embeddings = outputs.last_hidden_state.mean(dim=1).numpy()

# Normalize embeddings
normalized_embeddings = embeddings / np.linalg.norm(embeddings, axis=1, k

# Create FAISS index
d = normalized_embeddings.shape[1]  # Dimension of embeddings
index = faiss.IndexFlatIP(d)  # Index type
index.add(normalized_embeddings)
```

In [ ]:
```
query_embedding = normalized_embeddings[0]  # Use the first embedding as
k = 5  # Number of nearest neighbors to retrieve
distances, indices = index.search(np.expand_dims(query_embedding, 0), k)
print("Nearest neighbors:")
for i, idx in enumerate(indices[0]):
    print(f"Document {idx}: Similarity {1 - distances[0][i]}")
```

```
Nearest neighbors:
Document 0: Similarity 2.980232238769531e-07
Document -1: Similarity 3.4028234663852886e+38
Document -1: Similarity 3.4028234663852886e+38
Document -1: Similarity 3.4028234663852886e+38
Document -1: Similarity 3.4028234663852886e+38
```

OR

In [ ]:
```python
import numpy as np
from youtube_transcript_api import YouTubeTranscriptApi
from transformers import AutoTokenizer, AutoModel
import faiss

# Assuming you have a list of embeddings called 'embeddings'
# Convert embeddings to numpy array
embeddings_np = np.array(embeddings)

# Normalize embeddings (L2 normalization)
faiss.normalize_L2(embeddings_np)

# Create an index
index = faiss.IndexFlatIP(embeddings_np.shape[1])  # Index for inner prod
index.add(embeddings_np)

# Define a query embedding
query_embedding = np.random.rand(embeddings_np.shape[1]).astype(np.float3
query_embedding /= np.linalg.norm(query_embedding)

# Perform a k-nearest neighbors search
k = 5  # Number of nearest neighbors to retrieve
D, I = index.search(query_embedding.reshape(1, -1), k)

# Print the nearest neighbors and their distances
for i in range(k):
    print(f"Nearest Neighbor {i + 1}: Index {I[0][i]}, Distance {D[0][i]}
```

```
Nearest Neighbor 1: Index 0, Distance -0.05963487550616264
Nearest Neighbor 2: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 3: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 4: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 5: Index -1, Distance -3.4028234663852886e+38
```

クエリエンジンの作成

In [ ]:
```python
def search_query(query_embedding, index, top_k=5):
    D, I = index.search(query_embedding, top_k)
    return D, I
```

質問応答の命令

In [ ]:
```python
# Perform a k-nearest neighbors search
k = 5  # Number of nearest neighbors to retrieve
D, I = index.search(query_embedding.reshape(1, -1), k)

# Print the nearest neighbors and their distances
for i in range(k):
    print(f"Nearest Neighbor {i + 1}: Index {I[0][i]}, Distance {D[0][i]}

# Retrieve relevant documents using the indices I
relevant_documents = [documents[i] for i in I[0]]

# Print the relevant documents
for i, doc in enumerate(relevant_documents):
    print(f"Relevant Document {i + 1}: {doc}")

# Query the model for an answer
```

```
query = "what does this video like to say?"
response = query_engine.query(query)
print("Response:", response)
```

Nearest Neighbor 1: Index 0, Distance -0.05963487923145294
Nearest Neighbor 2: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 3: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 4: Index -1, Distance -3.4028234663852886e+38
Nearest Neighbor 5: Index -1, Distance -3.4028234663852886e+38
Relevant Document 1: GPT-4 is the latest AI system from OpenAI. The lab th
at created Dall-E. And ChatGPT. GPT-4 is a breakthrough in
problem solving capabilities. For example, you can ask it how
you would clean the inside of a tank filled with piranhas. And it'll give
you something useful. It can also read, analyze, or
generate up to 25,000 words of text. It can write code in all
major programming languages. And it understands images as input, and can r
eason with them
in sophisticated ways. Most importantly, after we created GPT-4,
we spent months making it safer and more aligned with how you want to use
it. The methods we've developed to
continuously improve GPT-4 will help us as we work towards AI
systems that will empower us all.
Relevant Document 2: GPT-4 is the latest AI system from OpenAI. The lab th
at created Dall-E. And ChatGPT. GPT-4 is a breakthrough in
problem solving capabilities. For example, you can ask it how
you would clean the inside of a tank filled with piranhas. And it'll give
you something useful. It can also read, analyze, or
generate up to 25,000 words of text. It can write code in all
major programming languages. And it understands images as input, and can r
eason with them
in sophisticated ways. Most importantly, after we created GPT-4,
we spent months making it safer and more aligned with how you want to use
it. The methods we've developed to
continuously improve GPT-4 will help us as we work towards AI
systems that will empower us all.
Relevant Document 3: GPT-4 is the latest AI system from OpenAI. The lab th
at created Dall-E. And ChatGPT. GPT-4 is a breakthrough in
problem solving capabilities. For example, you can ask it how
you would clean the inside of a tank filled with piranhas. And it'll give
you something useful. It can also read, analyze, or
generate up to 25,000 words of text. It can write code in all
major programming languages. And it understands images as input, and can r
eason with them
in sophisticated ways. Most importantly, after we created GPT-4,
we spent months making it safer and more aligned with how you want to use
it. The methods we've developed to
continuously improve GPT-4 will help us as we work towards AI
systems that will empower us all.
Relevant Document 4: GPT-4 is the latest AI system from OpenAI. The lab th
at created Dall-E. And ChatGPT. GPT-4 is a breakthrough in
problem solving capabilities. For example, you can ask it how
you would clean the inside of a tank filled with piranhas. And it'll give
you something useful. It can also read, analyze, or
generate up to 25,000 words of text. It can write code in all
major programming languages. And it understands images as input, and can r
eason with them
in sophisticated ways. Most importantly, after we created GPT-4,
we spent months making it safer and more aligned with how you want to use
it. The methods we've developed to
continuously improve GPT-4 will help us as we work towards AI
systems that will empower us all.
Relevant Document 5: GPT-4 is the latest AI system from OpenAI. The lab th
at created Dall-E. And ChatGPT. GPT-4 is a breakthrough in
problem solving capabilities. For example, you can ask it how

```
you would clean the inside of a tank filled with piranhas. And it'll give
you something useful. It can also read, analyze, or
generate up to 25,000 words of text. It can write code in all
major programming languages. And it understands images as input, and can r
eason with them
in sophisticated ways. Most importantly, after we created GPT-4,
we spent months making it safer and more aligned with how you want to use
it. The methods we've developed to
continuously improve GPT-4 will help us as we work towards AI
systems that will empower us all.
Response: ['Although we cannot predict exactly what will happen, and of co
urse our current progress could hit a wall, we can articulate the principl
es we care about\xa0most:', 'A gradual transition gives people, policymake
rs, and institutions time to understand what's happening, personally exper
ience the benefits and downsides of these systems, adapt our economy, and
to put regulation in place. It also allows for society and AI to co-evolv
e, and for people collectively to figure out what they want while the stak
es are relatively\xa0low.', 'We currently believe the best way to successf
ully navigate AI deployment challenges is with a tight feedback loop of ra
pid learning and careful iteration. Society will face major questions abou
t what AI systems are allowed to do, how to combat bias, how to deal with
job displacement, and more. The optimal decisions will depend on the path
the technology takes, and like any new field, most expert predictions have
been wrong so far. This makes planning in a vacuum very difficult.[^planni
ng]', 'In particular, we think it's important that society agree on extrem
ely wide bounds of how AI can be used, but that within those bounds, indiv
idual users have a lot of discretion. Our eventual hope is that the instit
utions of the world agree on what these wide bounds should be; in the shor
ter term we plan to run experiments for external input. The institutions o
f the world will need to be strengthened with additional capabilities and
experience to be prepared for complex decisions about\xa0AGI.']
```

ベクトルデータベース

ベクトルデータベース

*Faissデータベース

https://github.com/facebookresearch/faiss

*Pineconeデータベース（有料/無料試しあり）

https://www.pinecone.io/

*Qdrantデータベース

https://qdrant.tech/

*Chromaデータベース

https://docs.trychroma.com/

*Milvusデータベース

https://milvus.io/

*Waviateデータベース

https://weaviate.io/

Faiss を使った質問応答の実践例

In [ ]:
```
!pip install -q llama-index
!pip install -q openai
!pip install -q transformers
!pip install -q accelerate
```

In [ ]:
```
from llama_index.llms import OpenAI
from llama_index import VectorStoreIndex, SimpleDirectoryReader
from IPython.display import Markdown, display
from llama_index import GPTVectorStoreIndex, StorageContext
from llama_index.vector_stores.faiss import FaissVectorStore
```

In [ ]:
```
import os
os.environ["OPENAI_API_KEY"] = "your_api_key"
```

In [ ]:
```
!pip install -q faiss-gpu
```

In [ ]:
```
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
!wget https://raw.githubusercontent.com/MDASH-shinshu/general_resources/m
```

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

In [ ]:
```
import faiss

faiss_index = faiss.IndexFlatL2(1536)
```

In [ ]:
```
documents = SimpleDirectoryReader("data").load_data()
```

In [ ]:
```
index = VectorStoreIndex.from_documents(documents)
```

In [ ]:
```
from llama_index import GPTVectorStoreIndex, StorageContext
from llama_index.vector_stores.faiss import FaissVectorStore

# インデックスの作成
vector_store = FaissVectorStore(faiss_index=faiss_index)
storage_context = StorageContext.from_defaults(vector_store=vector_store)
index = GPTVectorStoreIndex.from_documents(
    documents,
    storage_context=storage_context
)
```

In [ ]:
```
# クエリエンジンの作成
query_engine = index.as_query_engine()
```

In [ ]:
```
# 質問応答
print(query_engine.query("ミコの幼馴染の名前は?"))
```

Pinecone

Pineconeの特徴

項目	特徴
高速	数10億のデータがあり、クリエは高速

フレッシュ|データの追加・編集・削除時に、インデックスを動的更新| |フィルタリング|ベクトル検索とメタデータフィルタを組み合わせて、より関連性の高いデータをすばやく取得| |フルマネージド|開始、仕様、スケーリングが簡単でスムーズで安全|

利用料金:

７日間試し無料期間

詳細料金:

https://www.pinecone.io/pricing/

Pinecone利用登録およびAPIキー取得

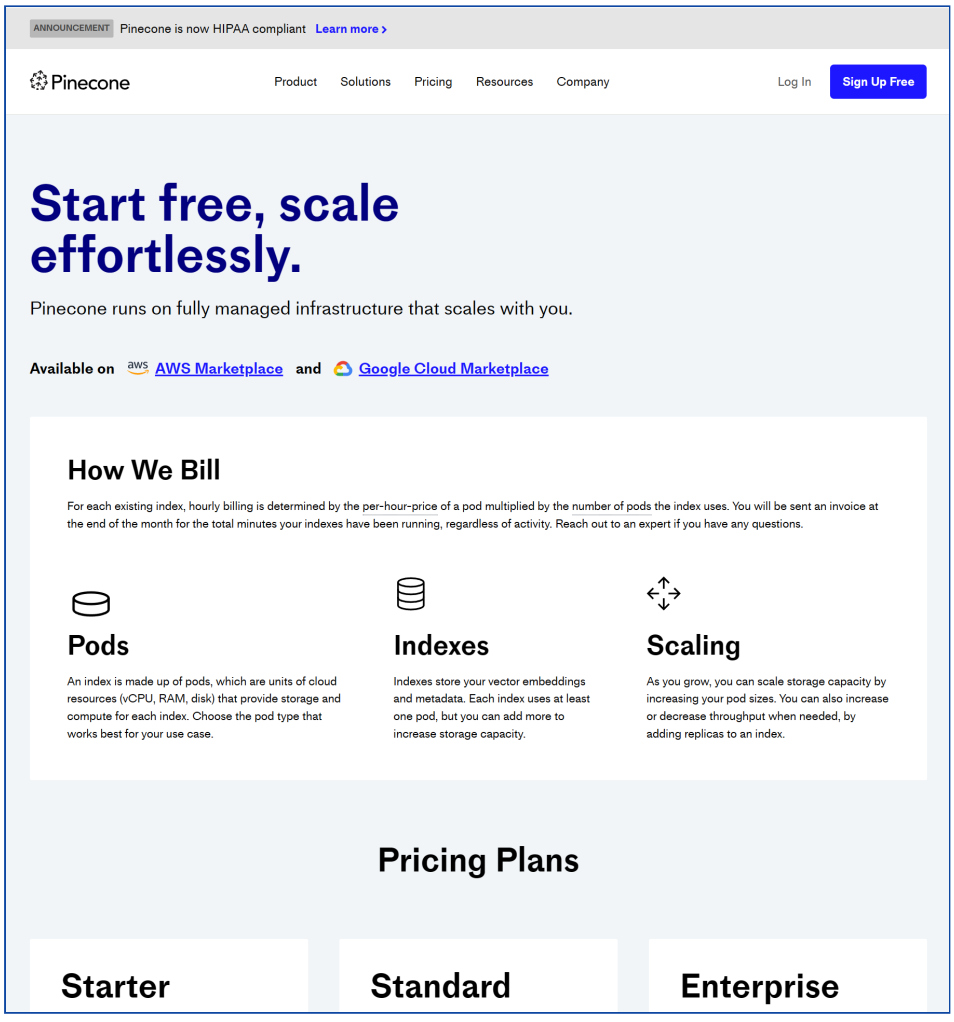


図5.12. 出所:Pinecone公式Webサイト |登録画面

Pinecone を使った質問応答の実践例

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

In [ ]:
```
from llama_index import SimpleDirectoryReader

# ドキュメントの読み込み（dataフォルダにドキュメントを配置しておきます）
documents = SimpleDirectoryReader("data").load_data()
```

In [ ]:
```
import pinecone

# pinecone-clientのインデックスの生成
api_key = "<PineconeのAPIキー>"
pinecone.init(api_key=api_key, environment="us-west1-gcp")
pinecone.create_index(
    "quickstart",
    dimension=1536,
    metric="dotproduct",
    pod_type="p1"
)
pinecone_index = pinecone.Index("quickstart")
```

In [ ]:
```
from llama_index import GPTVectorStoreIndex, StorageContext
from llama_index.vector_stores.pinecone import PineconeVectorStore

# インデックスの作成
vector_store = PineconeVectorStore(pinecone_index=pinecone_index)
storage_context = StorageContext.from_defaults(vector_store=vector_store)
index = GPTVectorStoreIndex.from_documents(
    documents,
    storage_context=storage_context
)
```

In [ ]:
```
# クエリエンジンの作成
query_engine = index.as_query_engine()
```

In [ ]:
```
# 質問応答
print(query_engine.query("ミコの幼馴染の名前は?"))
```