Pinecone＋Rag+OpenAIで作るFAQ構築セット取扱説明書

Pineconeとは

Pineconeは、文書やFAQを数値ベクトルとして保存・検索できるクラウド型のベクトルDBです。
本システムでは、ユーザーの質問に対してOpenAIの埋め込みモデルを使ってベクトル化を行い、
そのベクトルをPineconeで検索し、最も類似する情報を基に的確な回答を生成します。

ベクトルDBとは、文章・画像・音声などから抽出された特徴量（埋め込みベクトル）を
格納し意味的な距離による類似検索を効率的に行う事が出来ます。
従来のRDBがキーワード一致検索に強いのに対し、ベクトルDBは意味的な距離を高速に
計算できるという点に特徴があります。

Ragとは

RAG（Retrieval-Augmented Generation）は、検索の元となるデータを事前にベクトル化し
ユーザーの質問に対して類似する情報を検索してから回答を生成する仕組みで、本システムの様な
FAQ応答に最適です。

1. Rag関連のPinecone＋OpenAIの主な仕様

ここでは最低限知っておいた方がよい仕様のみ列挙し、詳細については割愛します。

1-1. Index

ベクトル検索の基本単位であり、Index 内に格納されるベクトルはすべて同じ次元数で統一されます。

1-2. Namespace

Index 内の論理的な区分け
顧客ごとや用途ごとにデータを分離できる
同じ ID でも Namespace が異なれば別データとして扱われる

1-3. Metric

類似度の計算方式を定義する
cosine: 方向性を重視
dotproduct: 内積を利用
euclidean: ユークリッド距離

1-4. Dimensions

ベクトルが持つ特徴量の数
Index 作成時に固定される
登録されるベクトルは全てこの次元数に一致する必要がある

1-5. 操作機能とエンドポイントURLの構成要素

① Pineconeエンドポイントの構成

https://<Index>-<プロジェクトID>.<サービス種別>.<リージョンコード>.pinecone.io

例）https://urata-soft-abc12de.svc.us-east-1.pinecone.io

② 操作機能とURL構成の対応

ベクトルのメタデータを類似検索する時は以下の様に指定します。

例）https://urata-soft-abc12de.svc.us-east-1.pinecone.io/query

操作機能一覧

操作機能	URL
ベクトル登録・更新	/vectors/upsert
類似検索	/query
ベクトルID削除	/vectors/delete
ベクトルとメタデータ取得	/vectors/fetch?id=<ID>
ベクトルのメタデータ更新	/vectors/update

1-6. Metadata

各ベクトルに属性情報を付与可能（例: タイトル、タグ、日付）

検索時にフィルタ条件として利用できる

1-7. OpenAIの主な埋め込みモデル

埋め込みモデルとは、テキストの意味を数値ベクトルに変換するAIのモデルの事です。

モデル名	特徴	トークン数上限
text-embedding-3-small	最新・軽量・高精度の主力モデル（2023末以降）	8191 tokens
text-embedding-3-large	より高精度・高コスト・主に高負荷分析用	8191 tokens
text-embedding-ada-002	旧主力モデル。高速で安価。精度はやや劣る	8191 tokens

2 . 環境構築

まずパッケージを解凍すると以下の様な構成になっていますので適宜、配置して下さい。

```
├── Flask
│   ├── .env                    APIキー、環境変数設定ファイル
│   ├── PDF
│   │   └── sample_specification.pdf      FAQ生成対象の元文書
│   ├── app.py                  Flask本体、Pinecone検索・OpenAI応答処理
│   ├── config.py               APIキーや環境変数の読込み処理
│   └── templates
│       └── index.html
├── README.md
├── config.env.template         APIキー、環境変数設定（POC検証用）
├── docs
│   ├── operating_instructions.pdf      ユーザー向け操作マニュアル
├── query_embeddings.py         クエリ文字列でPineconeへ検索処理
├── requirements.txt            Python ライブラリを一括でインストール設定ファイル
└── upload_embeddings.py        文書をベクトル化、Pineconeに登録処理
```

2-1. Pythonインストール

本システムではPythonを使用しますが、これを推す理由として以下があげられます。

・PineconeでのベクトルDB操作において公式 Python SDK が提供されておりベクトルの
　upsert、query、fetch 等すべての操作が簡潔に記述可能

・RAG構成全体のパイプライン統合
RAGテンプレートが Pythonで書かれており、フレームワークも充実
LangChain等を使えば、チャンク化→埋め込み→ベクトルDB登録→クエリ応答までが
シームレスに構築可能
・OpenAIライブラリでの生成＋埋め込み処理が一貫して行えAPIレスポンスのJSON操作もシンプル

① Microsoft Store からインストール（推奨）
start mswindowsstore://pdp/?productid=9PJPW5LDXLZ5
↑このコマンドを実行すると、Microsoft Storeで「Python 3.10」が開きます。
「インストール」ボタンを押すだけで完了です。

② Chocolatey 経由でインストール（開発者向け）
SetExecutionPolicy Bypass Scope Process Force;
`iex ((NewObject System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
choco install python version=3.10.9 y

③【Linux編】Ubuntu / Debian系
Python 3.10 / 3.11 / 3.12 インストール（任意で選択）
sudo apt update
sudo apt install y softwarepropertiescommon
sudo addaptrepository ppa:deadsnakes/ppa
sudo apt update
sudo apt install y python3.12 python3.12venv python3.12dev

④【Linux編】CentOS / RHEL 7, 8, 9系
Python 3.6〜3.9（標準レポジトリ or EPELから）
sudo yum update y
sudo yum install y epelrelease
sudo yum install y python3

⑤ インストール確認コマンド
python version
例： Python 3.10.9 と表示されればOKです。

2-2. OpenAI APIキー取得手順
https://platform.openai.com/signup にアクセスし、アカウントを作成またはログイン。
ログイン後、右上のプロフィールアイコン →「View API keys」を選択。
「Create new secret key」ボタンを押してキーを生成、コピーして保管。

2-3. Pinecone APIキー取得手順
https://www.pinecone.io/start/ にアクセスし、アカウントを作成またはログイン。
ログイン後、ダッシュボードに入り「API Keys」セクションへ移動。
「Create API Key」ボタンを押して名前を入力し、キーを生成、コピーして保管。

2-4.config.env.templateやFlask/.envファイル に以下の様に設定しておきます。
OPENAI_API_KEY=<2-2で取得したOpenAIのAPIキー>

PINECONE_API_KEY=<2-3で取得したPineconeのAPIキー>
PINECONE_URL=<PineconeエンドポイントURL ※1-5参照>
PINECONE_INDEX_NAME=<pineconeのindex名>

2-5. requirements.txtの作成

このシステムを構築する際に必要な Python ライブラリを一括インストールする設定ファイルです。
以下のコマンドでこれらのライブラリを自動インストール出来るので便利です。
> pip install -r requirements.txt

以下の様に初期設定していますが、OS等のバージョンアップ時には必要時、適宜変更して下さい。
openai>=1.2.0
pineconeclient>=3.0.0
tiktoken>=0.5.1
PyMuPDF>=1.23.0
pythondotenv>=1.0.0
pdfplumber==0.10.2
python-docx==1.1.0

2-7. セキュリティに関する注意事項

本システムでは最初に動きを検証するPOCとして、 config.txt から APIキー等を読み込む
形式にしていますが、この方法には**セキュリティ上の懸念**があります。
config.txt は HTML・JSと同一ディレクトリに置かれるため、Web公開領域に
含まれやすく、ブラウザで **F12 キーの**デベロッパーツールの**ネットワークやソースから APIキーが見えて**しまいます。
その為、次章の4-2でWeb公開時には、.envファイルにAPIキーを設定する方法を解説しています。

3. 各プログラムの詳細

同梱された各プログラムの用途は以下の通りです。
尚、必須パラメータに関しては極力、外部ファイルから取得したりパラメータで渡すようにしていますが
用途が明確で一貫性が求められるもののなかには 5 章の様にコード中に固定で設定している
ものもあります。カスタマイズする際はコードを適宜編集してください。

ファイル名	説明
upload_embeddings.py	PDFやテキスト等のナレッジファイルを読み込み、OpenAIの埋め込みモデルでベクトル化し、Pineconeに登録（upsert）するスクリプトで、初回のインデックス構築や文書更新時に実行
query_embeddings.py	ユーザーの入力テキストをOpenAI埋め込みモデルでベクトル化し、Pineconeから類似文書を検索・取得するスクリプトで、FAQボットの検索処理をする
config.py	.env からOpenAI APIキー・Pinecone APIキー・エンドポイントURLなどを読み込み、アプリ全体で共通の設定をここから参照する

app.py	Flaskのメインアプリ。 /query エンドポイントなどを定義し、query_embeddings.pyやOpenAI APIを呼び出して応答を返すサーバーサイド処理

4. ユースケースから実際の動きを検証（POC編）

あなたの会社はモバイルバッテリーを製造・販売しているとします。
そこでよくある質問（FAQ）を自動で回答できるAIヘルプボットをWebページに持ちたいと考えています。そのためにはまず同梱のsample_specification.txtを学習させて、実際にユーザから「バッテリーの容量は？」という質問に答えられるように構築してみましょう。

4-1. pineconeに「specification.txt」を読み込ませます

本システムではユーザーの質問文をベクトル化し、登録済みのテキストやPDFといったテキスト文書と照合して最も関連性の高い情報を抽出します。
手元にある「specification.txt」からベクトルを生成して、pineconeにアップロードします。
※ 読込ませるファイル名は英語名にして下さい。pinecone の仕様ではVector IDはASCII文字と指定されており、ファイル名からVector IDを取得している本システムでは日本語のファイル名にするとエラーとなります。
upload_embeddings.pyを、第１パラメータにフォルダを、題２パラメータにnamespaceを指定して実行します。
例）> python upload_embeddings.py ./PDF <namespace>
例）[処理開始] PDF/sample_specification.pdf
例）[成功] アップロード: sample_specification.pdf-chunk-1

4-2. 実際に質問してみましょう

「バッテリーの容量は？」という質問を query_embeddings.py にパラメータで渡すと4-1で登録した specification.txt の内容と最も関連性の高い情報を Pinecone から抽出し、OpenAI にその情報を渡して自然言語で回答してもらいます。
query_embeddings.pyを、第1パラメータに質問を、第2パラメータにnamespaceをして、以下のように1行で実行出来ます。
例）> python -c "from query_embeddings import ask_direct_answer;
print(ask_direct_answer('バッテリー容量は?', '<namespace>'))"
以下の様に回答を返せば成功です。
例）バッテリー容量は10,000mAhです。

4-3. Webページから質問してみましょう

今度は実用的なWebページを使って、質問を入力・送信しAIから自然言語の回答を得る処理をやってみます。
先ず2-4でconfig.env.templateに設定したAPIキー等の内容をセキュリティ対策として.envファイルに設定します。
また、質問を受取り処理するREST API構築にはPythonのフレームワークFlaskを採用しました。
FlaskはURLルーティングやAPIエンドポイントの定義が簡単に記述出来て依存ライブラリも軽量、数行のコードでAPIを定義できるのPythonのフレームワークです。

② 操作手順

1. ターミナルを開き、Flask/app.py があるディレクトリで以下のコマンドを実行します。
> python app.py

2. ブラウザで次のURLにアクセスします。

http://localhost:5000

3. 表示されたページの質問フォームに「バッテリーの容量は？」と入力し送信をクリック。

4. 検索された内容にOpenAIが自然言語で回答します。

③ 実行結果

以下の様に回答を返せば成功です。

5. パラメータ調整　よりよい回答のために

5-1. Pinecone 側の必須パラメータ調整

パラメータ	用途	推奨値
index	ベクトル検索対象となる物理的なデータストア単位	
namespace（※プログラム実行時のパラメータで指定）	同一index内のデータを論理的にグループ分けするラベル	
top_k	類似文書の取得件数	3〜5

5-2. OpenAI 側の主なパラメータ調整（modelはいずれも必須です）

パラメータ	用途	推奨値
model（Embedding Model）	テキストを数値ベクトルに変換	text-embedding-3-small
model（Chat Model）	自然言語の応答を生成	gpt-4o
temperature	ランダム性	0〜0.5
max_tokens	出力最大トークン数	200〜500
System Prompt	モデルへの役割・制約指示	用途に応じ設定
response_format	出力形式の制御	JSON推奨