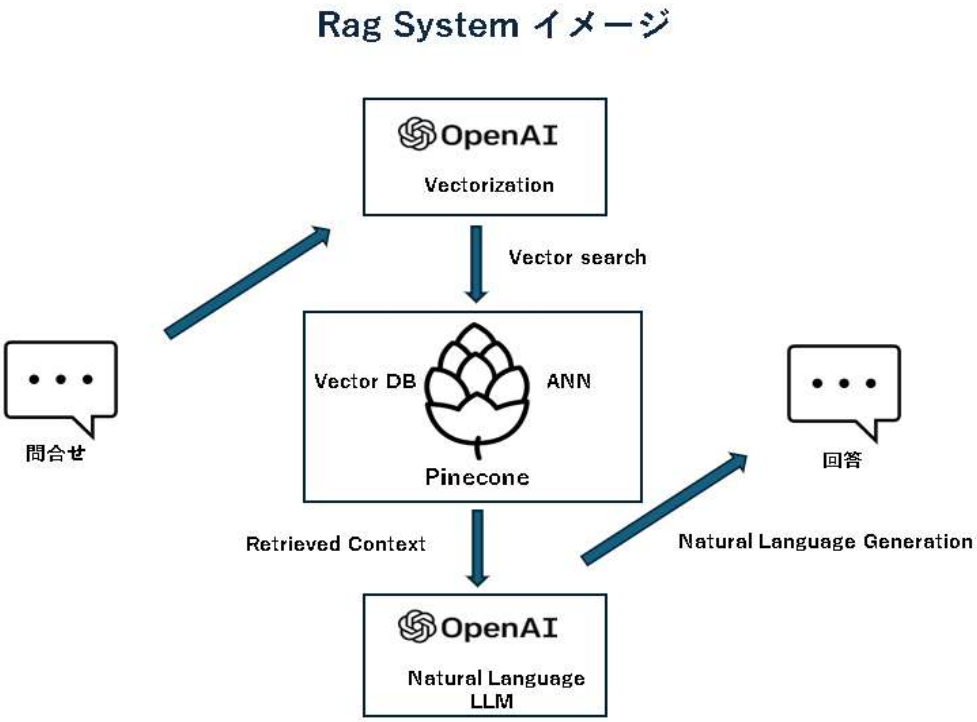
Pinecone＋Python＋OpenAIで作るRAG実戦ガイド

はじめに
本書はPineconeとOpenAI API、Pythonを活用して構築する検索拡張型のAI応答システムの実用的なRAGシステム導入ガイドです。
冗長な説明や抽象的な表現は極力排除し、現場で即使えることを第一に実行可能なスクリプト及び設定ファイルや構築手順を明解に記載していますので読むというより実行することで身につく一冊となっています。
前処理から埋め込み生成、検索・応答連携まで工程を網羅し、コマンドやコードにはその場で理解できるコメントを添えて解説していますので技術者でなくても扱いやすい構成となっています。

RAGシステムの全体概要図



Pineconeとは
Pineconeは2022年にニューヨークで創業された、クラウド型のベクトルDBサービスを提供する企業です。
検索精度とスケーラビリティに特化し、生成AIやRAG構成の中核技術として急速に普及しており開発者向けにもAPIベースで高速類似検索機能を提供しています。

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ので曖昧な表現や多様な質問にも柔軟に対応可能です。

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て
本システムは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す。
テキストによる問合せからOpenAIを通してベクトル化し、PineconeにベクトルDBを検索して結果を
OpenAIに返し自然言語処理によるテキストとして回答を返す仕組みです。

1. Pineconeと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。
尚、**この仕様は2025年9月執筆時のもの**です。最新のものについては公式ドキュメントを必ずご確認のうえ
必要に応じて本書のコードや設定内容を適宜アップデートしてください。

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
アップロード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. 機能毎のエンドポイントURL
1-5-1. 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>.<リージョンスクリプト>.pinecone.io
例）https://urata-soft-abc12de.svc.us-east-2.pinecone.io

1-5-2. 操作機能別のURL対応
ベクトルDBを操作する時は以下の様にエンドポイントの末尾に指定します。
操作機能一覧

操作機能	URL
ベクトルアップロード・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

各ベクトルに対して以下の表の様に「付加情報（属性）」を自由に持たせることができます。
これにより、検索時に特定の条件でフィルタリングを行うことが可能になります。

キー名	値（例）
"title"	"製品仕様書"
"category"	"FAQ"
"lang"	"ja"
"version"	"v2.1"
"tag"	["バッテリー", "充電", "仕様"]

これらの情報を付けておくことで、「特定カテゴリだけを対象に検索したい」
「英語のデータだけを除外したい」など、高度な絞り込みがPinecone側で簡単に
実行できるようになります。
特に、RAG構成では検索精度の向上や誤回答の抑制に重要な役割を果たします。
実用面でも、検索対象を明確に制御できるため、ナレッジの粒度管理にも有効です。

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	高精度・高コスト・高負荷分析用	8191 tokens
text-embedding-ada-002	旧主力モデル。高速で安価。	8191 tokens

2 . 開発環境構築手順

以下からダウンロード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。
https://github.com/NextGenAI-corder/Pinecone_Rag_OpenAI/tree/main/Pinecone_Rag_OpenAI

```
├── Flask
│   ├── .env                    APIキー、環境変数設定ファイル
│   ├── 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
├── PDF
│   └── sample_specification.pdf      FAQの元文書
├── 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から検索する処理
├── 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にアップロードする処理
```

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アップロード→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

Pythonのインストール方法としては以下が挙げられます。

2-1-1. Microsoft Store から

　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。

2-1-2. Chocolatey 経由で

　SetExecutionPolicy Bypass Scope Process Force;
　`iex ((NewObject System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))
　choco install python version=3.10.9 y

2-1-3. 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

2-1-4. Linux 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

2-1-5. インストール確認

　python --version
　例) Python 3.12.1 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及び.env ファイル設定

以下の様にスクリプトで取得するAPIキーなどをこのファイルから取得出来るよう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 ※1-5-1参照>
PINECONE_INDEX_NAME=<pineconeのindex名 ※1-1参照>

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
pinecone>=3.0.0
tiktoken>=0.5.1
PyMuPDF>=1.23.0
python-dotenv>=1.0.0
pdfplumber==0.10.2
python-docx==1.1.0
Flask>=2.0.0
```

2-6. 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として簡単にconfig.txt から APIキー等を読み込む
方法を採用していますが、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やJS等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、あとの4-3ではWeb公開時に、.envファイルにAPIキーを設定する方法を解説しています。

2-7. FAQ元のサンプル文書

サンプル元ネタファイル sample_specification.pdf は以下の様に書いておきます。（テキストファイルでも可）
本製品は、USB-AおよびUSB-C出力に対応したリチウムイオン式モバイルバッテリーです。
最大出力は20Wで、スマートフォン、タブレット、イヤホンなど様々なデバイスに急速充電が可能です。
バッテリー容量は10,000mAhで、iPhoneなら約2回分の充電が行えます。 動作温度・保管温度範囲は0°C〜40°Cの
範囲内で使用・保管し、45°C以上の高温下には絶対に放置しないようにして下さまた防水ではありませんので
安全性を重視し、過電流・過充電・過熱保護機能を搭載。残量確認用LEDインジケーター付き。
本製品は、USB-Aポートおよび USB-Cポートを搭載しており、対応ケーブルとしてUSB Type-Cです。

3. 各スクリプトの用途

同梱された各スクリプト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。
用途が明確で一貫性が求められるもののなかには５章のスクリプトの様にスクリプト中に
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スクリプト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用の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、初回のインデックス構築や文書更新時に実行する
query_embeddings.py	Webシステム以外に使用し、本件では動きを検証する為のPOC用に提供している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・取得し、FAQボットの検索処理をする
config.py	APIキー等を設定した .envファイルから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参照する
app.py	Flaskのメインアプリ。

4. ユースケースでRAGシステムの動きを理解する

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.pdf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に元ネタを登録する

本システムではユーザーの質問文をベクトル化し、アップロード済みのテキストや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。
sample_specification.pdf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、第２パラメータにnamespaceを指定
して実行します。
例）> python upload_embeddings.py "<ここに読み込むフォルダのパスを入力（例: ./PDF）>"
"<Pineconeのnamespaceをここに設定>"
例）[処理開始] PDF/sample_specification.pdf[成功] アップロード: sample_specification.pdf-chunk-1
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でアップロードした 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?', '"<Pineconeのnamespaceをここに設定>"'))"

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

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、数行のスクリプトでAPIを定義できるのPythonのフレームワークです。

4-3-1. 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 "<Pineconeのnamespaceをここに設定>"

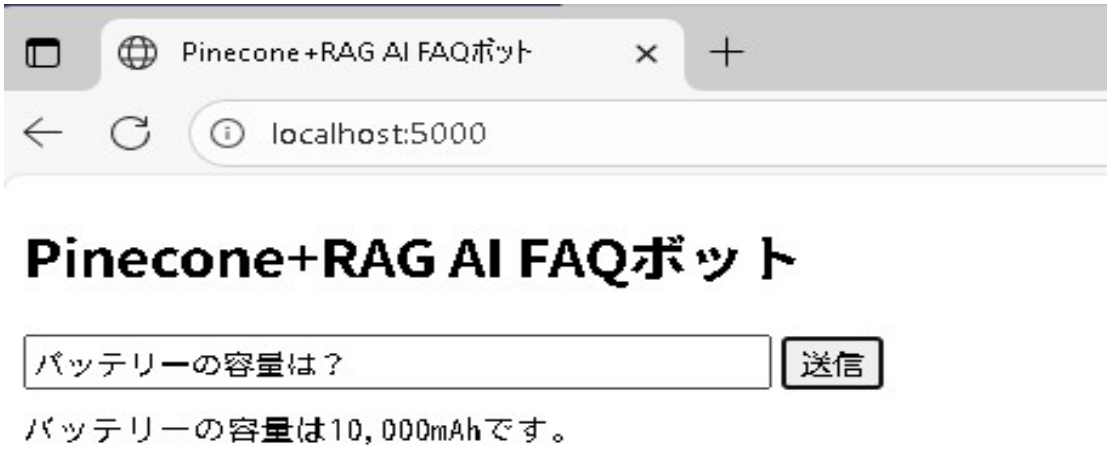
```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 963-682-455
127.0.0.1 - - [16/Oct/2025 05:45:16] "GET / HTTP/1.1" 200 -
```

※ 上記の様にWARNINGが出た場合ですが、これはFlaskの開発用サーバ（flask run や app.run()）が
本番用ではないのでセキュリティ、安定性、同時接続処理などに制限あるという警告で本番環境では
WSGIサーバー（Web Server Gateway Interfaceサーバー）を推奨しておりここでは問題ありません。

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。

4-3-2. 実行結果

以下の様に回答を返せば成功です。



5. より良い回答のためのパラメータ調整

5-1. Pinecone 側の必須パラメータ

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（※スクリプト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

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

※ modelはいずれも必須です