FAQ building kit instruction manual using Pinecone + Rag + OpenAI

What is Pinecone?

Pinecone is a cloud-based vector database that allows users to store and search documents and FAQs as numerical vectors. In this system, OpenAI's embedding model is used to vectorize user questions, and the vectors are searched in Pinecone to generate accurate answers based on the most similar information.

A vector database stores features (embedding vectors) extracted from text, images, audio, etc., and enables efficient similarity searches based on semantic distance. While traditional RDBs are strong in keyword matching searches, a vector database is unique in that it can quickly calculate semantic distance.

What is Rag? RAG (Retrieval-Augmented Generation) is a system that pre-vectorizes the data that will be used for searches, searches for similar information to the user's question, and then generates an answer, making it ideal for FAQ responses like this system.

1. Main specifications of Rag-related Pinecone and OpenAI

Here we will only list the specifications that you should be aware of and omit the details.

1-1. Index

It is the basic unit of vector search, and all vectors stored in the index have the same number of dimensions.

1-2. Logical division within the Namespace Index

Data can be separated by customer or purpose. The same ID will be treated as separate data if the Namespace is different.

Metric

Define the calculation method for similarity cosine: Emphasizes direction dotproduct: Uses the dot product euclidean: Euclidean distance

1-4. Dimensions

The number of features in the vector This is fixed when creating an index. All registered vectors must match this number of dimensions.

1-5. Operation Functions and Components of Endpoint URL

ÿ Pinecone endpoint configuration

https://<Index>-<Project ID>.<Service Type>.<Region Code>.pinecone.io

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

ÿ Operation functions and URL structure

When searching for similarity in vector metadata, specify as follows:

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

List of operation functions

Operation	URL
functions Vector	/vectors/upsert
registration/	/query
update Similarity	/vectors/delete
search Delete vector ID Get vector and metadata	/vectors/fetch?id=<ID>
Vector metadata update	/vectors/update

1-6. Metadata

Each vector can have attribute information attached (e.g. title, tags, date)

Can be used as a filter condition when searching

1-7. OpenAI's main embedding models

An embedding model is an AI model that converts the meaning of text into a numerical vector.

Model name		Token limit
text-embedding-3-small	Features The latest, lightweight, and highly accurate flagship model (After the end of 2023)	8191 tokens
text-embedding-3-large	Higher precision, higher cost, and mainly high load For analysis	8191 tokens
text-embedding-ada-002	Old flagship model. Fast and cheap. Accuracy is or inferior	8191 tokens

2. Environment construction

First, unzip the package and it will have the following structure, so please place it as appropriate.

ÿÿÿ Flask

ÿ ÿÿÿ .env ÿ ÿÿÿ API key, environment variable setting file

app.py ÿ ÿÿÿ config.py Flask itself, Pinecone search and OpenAI response processing

ÿ ÿÿÿ templates Reading API keys and environment variables

ÿ ÿÿÿ index.html

ÿÿÿ PDF

ÿ ÿÿÿ sample_specification.pdf Original FAQ document

ÿÿÿ README.md

ÿÿÿ config.env.template API key, environment variable settings (for POC verification)

ÿÿÿ docs

ÿ ÿÿÿ operating_instructions.pdf User operation manual

ÿÿÿ query_embeddings.py Searches Pinecone using query strings

ÿÿÿ requirements.txt Install Python libraries in bulk with configuration files

ÿÿÿ upload_embeddings.py: Process to vectorize documents and register them in Pinecone

2-1. Install Python

This system uses Python, and the reasons for this are as follows:

•Pinecone provides an official Python SDK for vector DB operations.

All operations such as upsert, query, and fetch can be written succinctly.

•Pipeline integration of the entire RAG configuration

RAG templates are written in Python and have a rich framework

By using LangChain etc., you can seamlessly build chunking ÿ embedding ÿ vector DB registration

ÿ query response.

•Generation and embedding processes can be performed consistently using the OpenAI library, and JSON manipulation of API responses is simple.

ÿ Install from the Microsoft Store (recommended)

start mswindowsstore://pdp/?productid=9PJPW5LDXLZ5 ÿRunning

this command will open "Python 3.10" in the Microsoft Store. Simply press the "Install"

button to complete the installation.

ÿ Install via Chocolatey (for developers)

SetExecutionPolicy Bypass Scope Process Force; `iex

((NewObject System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1')) choco install python

version=3.10.9 y

ÿÿLinuxÿUbuntu/Debian

Install Python 3.10 / 3.11 / 3.12 (optional) 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

ÿ [Linux version] CentOS / RHEL 7, 8, 9 series

Python 3.6 to 3.9 (from standard repository or EPEL)

sudo yum update y

sudo yum install y epelrelease

sudo yum install y python3

ÿ Installation confirmation command

python version

Example: If it shows Python 3.10.9 then it's OK.

2-2. Procedure for obtaining an OpenAI API key

Go to https://platform.openai.com/signup and create an account or log in. After logging in, click

the profile icon in the top right corner and select "View API keys."

Click the "Create new secret key" button to generate a key, then copy and save it.

2-3. How to obtain a Pinecone API key

Go to https://www.pinecone.io/start/ and create an account or log in. After logging in, go to

the dashboard and go to the "API Keys" section.

Press the "Create API Key" button, enter a name, generate a key, copy it and keep it.

2-4.Set the following in the config.env.template or Flask/.env file.

OPENAI_API_KEY=<OpenAI API key obtained in 2-2>

PINECONE_API_KEY=<Pinecone API key obtained in 2-3>

PINECONE_URL=<Pinecone endpoint URL *See 1-5>

PINECONE_INDEX_NAME=<pinecone index name>

2-5. Create requirements.txt

This is a configuration file that installs all the Python libraries required to build this system. It is convenient because you can automatically install these libraries with the following command.

> pip install -r requirements.txt

The default settings are as follows, but please change them as necessary when upgrading your OS etc.:

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. Security Precautions

As an initial POC to verify the system's functionality, this system reads the API key and other information from config.txt , **but this method raises security concerns. Because config.txt is placed in the same directory as HTML and JS, it is easily included in the publicly accessible web area,** making the API key visible **from the network and source options in the developer tools (F12 key)** in **the browser. For this reason, in the next chapter, 4-2, we explain how to set the API key in the .env file when publishing to the web.**

3. Details of each program

The purpose of each included program is as follows.

Note that we have tried to obtain required parameters from external files or pass them as parameters as much as possible, but some programs have fixed settings in the code as shown in Chapter 5, as their purpose is clear and consistency is required. Please edit the code as appropriate when customizing.

File name	explanation
upload_embeddings.py	This script reads knowledge files such as PDFs and text, vectorizes them using OpenAI's embedding model, and registers (upserts) them in Pinecone. It is executed when building the index for the first time or updating a document.
query_embeddings.py	The user's input text is vectorized using the OpenAI embedding model. A script that searches and retrieves similar documents from Pinecone, and processes the search for the FAQ bot
config.py	Read OpenAI API key, Pinecone API key, endpoint URL, etc. from .env and refer to common settings for the entire app from here.

app.py	The main Flask app. Server-side processing that defines the /query endpoint, calls query_embeddings.py and the OpenAI API, and returns a response

4. Verifying actual behavior from use cases (POC)

Let's say your company manufactures and sells mobile batteries. You want to have an AI help bot on your web page that can automatically answer frequently asked questions (FAQs). To do this, you first train the bot using the included sample_specifications.txt file and then ask actual questions from users. Let's build something that answers the question, "What is the capacity of the battery?"

4-1. Load "specification.txt" into pinecone

This system vectorizes the user's question and compares it with registered text documents such as text and PDFs to extract the most relevant information. Generate a vector from the "specification.txt" you have on hand and upload it to pinecone. *Please use English names for the file to be read. Pinecone's specifications specify that Vector IDs are ASCII characters, and this system obtains Vector IDs from filenames, so an error will occur if the filename is in Japanese. Execute upload_embeddings.py, specifying the folder as the first parameter and the namespace as the second parameter. Example: > python upload_embeddings.py ./PDF <namespace>

Example: [Start processing] PDF/sample_specification.pdf Example: [Success] Upload: sample_specification.pdf-chunk-1

4-2. Let's actually ask a question

When you pass the question "What is the battery capacity?" as a parameter to query_embeddings.py, it extracts the information most relevant to the content of specification.txt registered in 4-1 from Pinecone, Pass that information to OpenAI and have it answer in natural language. query_embeddings.py can be executed in one line as follows, with the question as the first parameter and the namespace as the second parameter. Example) > python -c "from query_embeddings import ask_direct_answer;

```
print(ask_direct_answer('What's the battery capacity?', '<namespace>'))"
```

The response is successful if it returns the following: Example: Battery capacity is 10,000mAh.

4-3. Ask a question from a web page

Next, we'll use a practical web page to input and submit a question and receive a natural language response from the AI.

First, we'll set the API key and other information set in config.env.template in 2-4 in the .env file as a security measure. We also used the Python framework Flask to build the REST API that receives and processes questions.

Flask is a Python framework that allows you to easily define URL routing and API endpoints, has lightweight dependencies, and allows you to define an API with just a few lines of code.

ÿ Operation procedure

1. Open a terminal and run the following command in the directory where Flask/app.py is located: > python app.py

2. Access the following URL in your browser:

http://localhost:5000

3. Enter "What is the battery capacity?" in the question form on the page that appears and click Submit.

4. OpenAI will respond to your search in natural language.

ÿ Execution results

If you reply as follows, it is successful.

5. Parameter adjustment for better answers

5-1. Adjusting the required parameters on Pinecone

Parameters	Purpose	Recommended value
index	Vector search target physical data Data store unit	
namespace (*Program execution parameters (Specify on the meter)	Logically group data in the same index Label to separate loops	
top_k	Number of similar documents obtained	3 to 5

5-2. Main parameter adjustments on the OpenAI side (all models are required)

Parameters	Purpose	Recommended value
model (Embedding Model) Convert text to a numeric vector text-embedding-3-small		
model (Chat Model) Generate natural language responses		gpt-4o
temperature	Randomness	0–0.5
max_tokens	Maximum number of output	200 to 500
System Prompt	tokens Role and constraint instructions	Set according to the purpose
response_format	for the model Output format control	JSON is recommended