ベクトルDB管理APIチュートリアル

はじめに

本書は、本サービスが提供するベクトルDB管理APIのチュートリアルを記載します。

また、本書の対象読者は以下の通りです。

- 本サービスと連携したシステムや製品開発を行う開発者
- 管理ポータルを使用せず、ベクトル検索DBのリソースを操作したいユーザ

なお、本書ではPythonのサンプルコードを記載しておりますが、本サービスのAPIはREST形式のため、他の言語からもご利用いただけます。

> - 本ガイドに記載する「https://<サーバのドメイン名>/」はGenerative AI FWがインストールされているサーバのドメイン名に置き換えてAPIを実行してください。
> - 本サービスではHTTPSに既定では自己証明書を使用しています。そのためAPI利用時に考慮が必要です。詳細は「スタートアップマニュアル（導入準備編）」をご確認ください。

ベクトルDB管理API チュートリアルの流れ

ベクトルDB管理APIでは、ベクトル検索DBに対して、インデックスの作成削除や、文書の登録削除を行う機能を有します。本チュートリアルでは、これらの一連の操作方法を記載します。

本チュートリアルの流れは以下です。

1. インデックスを作成する
2. （応用）requestsライブラリを利用したPythonプログラムで文書登録をする
3. （応用）LangChainライブラリを利用したPythonプログラムでチャンク文書を登録する
4. 登録した文書に対してチャンク検索を行う
5. 登録した文書に対して検索対話を行う
6. 登録した文書情報を取得する
7. 登録済みインデックス情報を取得する
8. 登録済みインデックス情報の変更を行う
9. 登録した文書情報を削除する
10. 作成したインデックスを削除する

インデックスを作成する

コマンド実行でベクトル検索DB上に新規にインデックスを作成することが出来ます。

作成されたインデックスにはグループやユーザが紐づいていないので、引き続きコマンドからインデックスの操作を行いたい際は、作成後に管理ポータルから作成したインデックスにAPI USER、もしくはAPI USERが所属するグループの紐づけを行ってください。

> - 上記のAPI USERとは事前にシステムに登録されているシステムユーザを指します。詳細は「管理ポータル操作ガイド（ユーザ登録編）」をご参照ください。
> - インデックスにグループやユーザを紐づける方法は、「管理ポータル操作ガイド（インデックス・文書管理編）」をご参照ください。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。

```
curl -X POST https://<サーバのドメイン名>/genai-search-api/index/createIndex ¥
-H "Content-Type: application/json" -H "Authorization: Bearer <API Key>" ¥
--data "<リクエストパラメータ>"
```

リクエストパラメータは必須のパラメータと任意のパラメータがあります。

パラメータ名	必須	説明

vectorIndex	○	作成するインデックス名を指定します。インデックス名には英数字(全角/半角)と日本語(全角/半角)とハイフン（-とー）のみ使用可能です（最初と最後にハイフンを使用することは不可）。 また文字数は2～64文字の範囲で指定が可能です。加えて、すでに作成済みのインデックス名を指定することはできません。
embeddingModel	-	文書登録される際のベクトル化モデルをインデックスに紐づけて登録するパラメータで、管理ポータルの利用可能な埋め込みモデル一覧のモデルIDから確認できます。
comment	-	インデックスに対する説明文などをインデックスに紐づけて登録できるパラメータです。登録した内容はインデックス一覧画面から確認できます。

以下はリクエストパラメータの例です。

```
{
  "vectorIndex": "test-index",
  "embeddingModel": "multilingual-e5-large",
  "comment" : "テスト用インデックス"
}
```

インデックスの作成に成功するとJson形式で作成されたインデックスの名前が返却されます。

```
{"vectorIndex":"test-index"}
```

（応用）requestsライブラリを利用したPythonプログラムで文書登録をする

文書登録には以下で紹介するpythonプログラムを利用することができます。

この章では、Base64形式にエンコードしたファイルをrequestsライブラリでAPIに送信して文書登録を行うPythonプログラム(以降、文書登録プログラム)を紹介します。

文書登録時の注意事項

- 登録先インデックスについて、事前にAPI USERが所属しているグループ、もしくはAPI USERがインデックスに紐づいていることをご確認ください。
- 拡張子が .txt や .csv のファイルを登録する際、内部で文字コードの特定ができず、登録エラーとなる場合があります。
  このような場合は、文字コードが判別しやすい形式（例：UTF-8 BOM付き） でファイルを保存し直したうえで、再度登録をお試しください。
- ファイル内に含まれる「文字情報」のみを読み込み・解析対象とし、埋め込み画像・スキャン画像・図表画像内の文字情報を直接認識することはできません。

requestsライブラリのインストール

文書登録プログラムでは、`requests`ライブラリが必要ですので、以下のコマンドでインストールを行ってください。

インストールコマンドの例

```
pip install requests tzdata
```

文書登録プログラムのダウンロード

以下からrequestsライブラリを使用して、インデックスへの文書登録を行うPythonファイルを参照できるため、ローカルのPython実行環境に同じ記載のプログラムを配置してください。

以下のリンクから該当のプログラムを参照できます。

RAG文書登録プログラム

文書登録プログラムの実行 - パラメータ詳細

配置したPythonプログラムを実行することで指定したインデックスへの文書登録ができます。

実行コマンドには以下のようなパラメータがあり、必須のパラメータを指定して実行して下さい。また任意のパラメータについては `--<任意のパラメータ名> 値` のように指定してください。

パラメータ名	必須	説明
api_url	○	文書登録APIのURL。https://<サーバのドメイン名>/+”/genai-search-api/document/addDocument”を指定してください。
auth_token	○	API認証に使用するトークン。APIキーを指定してください。
file_or_directory_path	○	処理対象のファイルまたはフォルダのパス • ファイルの場合：指定されたファイルを処理対象として読み込みます。 • フォルダの場合：指定したフォルダ配下のファイルを再帰的に探索し、全てのファイルを処理対象として読み込みます。 <データの送信方式について> • RAG文書登録プログラム ◦ 読み込んだファイルは base64エンコードされ、文書登録APIに送信されます。 • チャンク文書登録プログラム ◦ 読み込んだファイルは自動的にチャンク分割され、文書登録APIに送信されます。 ◦ 送信されるチャンクは以下のようなJson形式で構成されています： ▪ `page_content`：チャンクの中身（テキスト内容）。 ▪ `metadata`：チャンクに関連するメタデータ情報。 <APIに送信されるチャンクデータの例> `{ ”page_content”: ”ドキュメントの内容１”, ”metadata”: { ”source”: ”/aa/bb/cc.ppt”, ”page”: 0 }}`
vector_index	○	文書を登録するインデックス名
url	-	登録ファイルのURL。検索対話時にメタデータとして参照できるようになります。 file_or_directory_pathでフォルダパスを指定時は、file_or_directory_pathの値をベースURLとして、それぞれのファイルパスに階層ごとに別々のパスを付与して登録を行います。
overwrite	-	上書きフラグ（Trueの場合、既存のファイルを上書きをします。） デフォルトは上書きされます。

custom_metadata	-	任意のメタデータ（キーと値のペア）を指定できます。Json形式で指定し、検索対話時にメタデータとして参照できるようになります。
kwargs	-	文書登録時の追加オプション指定。 文書のチャンク化を実施する際のオプションを指定できます。 split_chunk_size ： テキストチャンクのサイズ（文章を分割するときのサイズ）で、0～512の範囲で選択可能。指定なしの場合は500で動作します。 split_overlap_size ： テキストチャンクのオーバラップサイズ（各チャンクの境界が重なる部分のサイズ）で0～split_chunk_sizeの指定値の範囲で選択可能。指定なしの場合は128で動作します。

以下、実行環境に応じたコマンド実行例になります。

実行コマンド(Linux)

```
python script_addDocuments.py <api_url> <auth_token> <directory_or_file_path> ¥
<vector_index> --url <base_url> --overwrite <overwrite> --custom_metadata <custom_data> ¥
--kwargs '{"split_chunk_size": "<chunk_size>", "split_overlap_size": "<overlap_size>"}'
```

Windows環境のコマンドプロンプトでは仕様上、ダブルクォーテーションをエスケープする必要があるため、--kwargsオプション指定時は以下を参考にしてください。

実行コマンド(Windows - コマンドプロンプト)

```
python script_addDocuments.py <api_url> <auth_token> <directory_or_file_path> ^
<vector_index> --url <base_url> --overwrite <overwrite> --custom_metadata <custom_data> ^
--kwargs "{¥"split_chunk_size¥": ¥"<chunk_size>¥", ¥"split_overlap_size¥": ¥"<overlap_size>¥"}"
```

登録に成功すると、以下のような実行結果が返却されます。

実行結果の例

```
Processing single file: D:¥test¥test.txt
Processed test.txt at 2024-07-23 11:54:43.013309+09:00: 200 null
```

（応用）LangChainライブラリを利用したpythonプログラムでチャンク文書を登録する

この章では、LangChainというライブラリを使用して文書を読み込み、扱いやすいサイズに分割したチャンクを用いて文書登録を行うPythonプログラム(以降、チャンク文書登録プログラム)を紹介します。

今回紹介するのは、テキストファイルの読み取りと分割のみに対応したプログラムです。 他の拡張子の文書の読み込み部分やチャンク分割部分は、実際のユースケースに合わせて適宜拡張してください。

LangChainとは？

LangChainは、自然言語処理（NLP）や生成AIを利用したアプリケーションを簡単に作成するためのPythonライブラリです。このライブラリは以下のような特徴があります：

- AIモデルとの連携が簡単： LangChainはOpenAIのGPTなどのAIモデルを活用しやすいように設計されています。
- 大規模な文書を処理する機能： 大量のテキストを効率的に処理し、AIでの応答生成や検索機能の実装が可能です。
- 「チャンク」と呼ばれる分割処理のサポート： 長い文書を自動的に分割し、AIモデルが扱いやすくする仕組みを提供しています。

チャンクとは？

チャンク（chunk）とは、長い文章や文書を小さな部分に分割したものです。

通常、アプリケーションのAPI内部で文書は自動的に分割されますが、ユーザが分割のルールや細かい設定を自由にカスタマイズしたい場合、この機能を利用することで、分割方法を支配的にコントロールできます。以下のような場合に役立ちます：

- **特定の分割ルールを指定したい場合**：
  - 文の意味が途切れないように段落ごとに分割したい。
  - 固定の文字数で分割したい。
- **分割後の内容を事前に確認したい場合**：
  - 分割後のチャンクが適切かを確認し、必要に応じて微調整したい。

必要なライブラリのインストール

以下で紹介するチャンク文書登録プログラムでは、requestsライブラリのほかに、LangChain関連のライブラリが必要です。以下のコマンドを実行してインストールを行ってください。

```
pip install langchain langchain-community requests chardet tiktoken tzdata transformers
```

チャンク文書登録プログラムのダウンロード

以下からrequestsライブラリを使用して、インデックスへの文書登録を行うpythonファイルを参照できるため、ローカルのPython実行環境に同じ記載のプログラムを配置してください。また、本プログラムはテキストファイルの読み込みに対応しております。別の拡張子のファイルを読み取って実行する必要がある場合は、お手数ですが、以下のプログラムを参考に改修を行っていただきますようお願いいたします。

以下のリンクから該当のプログラムを参照できます。

チャンク文書登録プログラム

また、実行時に関連ファイルが必要になるため、実行前に以下の作業を実施してください。

1．「intfloat--multilingual-e5-large-tokenizer」という名前のフォルダを作成し、チャンク文書登録プログラムと同じ階層に配置してください。

2．intfloat/multilingual-e5-large at main から以下4つのファイルをダウンロードして、上記のフォルダに格納してください。

・config.json

・special_tokens_map.json

・tokenizer_config.json

・tokenizer.json

チャンク文書登録プログラムの実行

配置したpythonプログラムを実行することで指定したインデックスへのチャンク文書登録ができます。

実行コマンドは以下で、パラメータについてはRAG文書登録プログラムと同じになりますので、詳細は前章の文書登録プログラムの実行 - パラメータ詳細をご覧ください。

実行コマンド(Linux)

```
python script_addChunks.py <api_url> <auth_token> <directory_or_file_path> ¥
<vector_index> --url <base_url> --overwrite <overwrite> --custom_metadata <custom_data> ¥
--kwargs '{"split_chunk_size": "<chunk_size>", "split_overlap_size": "<overlap_size>"}'
```

Windows環境のコマンドプロンプトでは仕様上、ダブルクォーテーションをエスケープする必要があるため、--kwargsオプション指定時は以下を参考にしてください。

実行コマンド(Windows - コマンドプロンプト)

```
python script_addChunks.py <api_url> <auth_token> <directory_or_file_path> ^
<vector_index> --url <base_url> --overwrite <overwrite> --custom_metadata <custom_data> ^
--kwargs "{¥"split_chunk_size¥": ¥"<chunk_size>¥", ¥"split_overlap_size¥": ¥"<overlap_size>¥"}"
```

登録に成功すると以下のような実行結果が返却されます。

実行結果の例

```
Processing single file: D:¥test¥test.txt
Processed test.txt at 2024-07-23 11:54:43.013309+09:00: 200 null
```

実行時に表示される警告メッセージについて

プログラムの実行時に、以下のような警告メッセージが表示される場合がありますが、問題なく動作するため、このメッセージは無視していただいて構いません(学習や推論用のフレームワークが見つからないことを示す警告メッセージです。)。

```
None of PyTorch, TensorFlow >= 2.0, or Flax have been found.
Models won't be available and only tokenizers, configuration and file/data utilities can be used.
```

登録した文書に対してチャンク検索を行う

以下のコマンド実行で、入力されたキーワードをもとに検索し、キーワードに関連するチャンク（文書の一部）を取得することができます。

検索対象は、パラメータで指定したインデックス、かつAPI USERが所属するグループ、もしくはAPI USERが紐づいてるものが対象になります。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。API Keyは管理ポータルの契約情報から確認できます。

```
curl -X POST https://<サーバのドメイン名>/genai-search-api/document/searchRelatedChunks/ ¥
-H "Content-Type: application/json" -H "Authorization: Bearer <API Key>" ¥
--data "<リクエストパラメータ>"
```

リクエストパラメータは必須のパラメータと任意のパラメータがあります。

リクエストパラメータの各項目の詳細は以下のとおりです。

パラメータ名	型	デフォルト	必須	説明
searchWord	string	-	○	検索する文字列
vectorIndex	string	-	△	検索対象となるベクトルストアのインデックス名を指定する。 vectorIndex、vectorIndexesのどちらかを指定する。
vectorIndexes	array[string]	-	△	検索対象となる複数のベクトルストアのインデックス名を指定する場合は配列で指定する。 vectorIndex、vectorIndexesのどちらかを指定する。
searchOption	object	{"searchType": "hybrid"}	-	ベクトル検索のオプションをJson形式で必要に応じて複数記載する
searchOption.searchType	string	hybrid	-	検索手法 similarity：類似度で検索する方式 hybrid：類似度検索とキーワード検索を組み合わせて検索する方式

searchOption.topK	int	4	-	返却するチャンク数の最大数 (最小値:1 最大値:100)

以下はリクエストパラメータの例です。

```
{
  "searchWord": "スナップショット",
  "vectorIndex": "test-index",
  "searchOption": {
    "searchType": "hybrid",
    "topK": 6
  }
}
```

検索に成功するとJson形式で関連するチャンクが関連度の高い順位に返却されます。

レスポンスボディの各項目の詳細は以下のとおりです。

パラメータ名	型	必須	説明
relatedChunks	array[object]	○	searchWordで指定された文章に近いチャンクの情報を返却する
relatedChunks[].score	float	○	マッチしたチャンクの関連度スコア
relatedChunks[].content	string	○	チャンク本文
relatedChunks[].metadata	object	○	チャンクのメタデータ
relatedChunks[].metadata.source	string	○	チャンク元のファイルパス
relatedChunks[].metadata.folder	string	○	インデックス名
relatedChunks[].metadata.insertDate	string	○	登録時間
relatedChunks[].metadata.url	string	○	ソース文書の参照URL(文書登録時に任意に指定できる)
relatedChunks[].metadata.page	int	-	ソース文書のページ番号
relatedChunks[].metadata.任意のメタデータ	Object	-	文書登録時に任意のメタデータを登録している場合、任意のメタデータを返却する。

以下はレスポンスの例です。

```
{
  "relatedChunks": [
    {
      "score": 0.9326973,
      "content": "インデックス合算の最大登録容量は100GBです。管理ポータルからの確認方法は…",
      "metadata": {
        "source": "guide_v2.pdf",
        "folder": "test-index",
        "insertDate": "2025-10-01T08:30:12Z",
        "url": "https://portal.example.com/docs/rag/guide_v2.pdf#page=12",
        "page": 12
      }
    },
    {
      "score": 0.917397,
      "content": "Snapshot retention の最小・最大保持数は…",
```

```
      "metadata": {
        "source": "operation.md",
        "folder": "test-index2",
        "insertDate": "2025-09-28T04:11:47Z",
        "url": ""
      }
    }
  ]
}
```

検索でヒットする文書数が指定した値より小さい場合、返却する数はtopKで指定した数より小さくなる場合があります。

登録した文書による検索対話を利用する

文書登録を行ったインデックスを選択して、検索対話を利用することができます。利用方法は、チャットUIから利用する場合と検索対話のAPIから利用する場合の2パターンがあります。

- チャットUIから検索対話を行いたい場合は「チャット画面利用ガイド」をご確認ください。
- APIから検索対話を行いたい場合は「検索対話チュートリアル」をご確認ください。

登録した文書情報を取得する

以下のコマンド実行でインデックスに登録済みの文書の情報を取得できます。

実行時は指定するインデックスについて、事前にAPI USERが所属しているグループ、もしくはAPI USERがインデックスに紐づいていることをご確認ください。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。インデックス名には、文書を登録したインデックス名を指定してください。

```
curl -X GET ¥
https://<サーバのドメイン名>/genai-search-api/document/listDocument?vectorIndex=<インデックス名> ¥
-H "Content-Type: application/json" -H "Authorization: Bearer <API Key>"
```

取得に成功すると、以下のようにJson形式で指定したインデックスの登録済みの文章情報の一覧(ファイル名、登録時に指定したurl(指定がなければ空)、登録日時)が返却されます。

```
{"documents":[
{"filepath":"file.pdf","url":"","insertDate":"2024-12-09T07:16:59.147Z"},
{"filepath":"test.txt","url":"https://example.com/test","insertDate":"2024-12-10T05:20:11.304Z"}]
}
```

登録済みインデックス情報を取得する

以下のコマンド実行で作成済みのインデックス情報を取得できます。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。

```
curl -X GET https://<サーバのドメイン名>/genai-search-api/index/listIndex ¥
-H "Content-Type: application/json" -H "Authorization: Bearer <API Key>"
```

取得に成功すると以下のようにJson形式で自分が所属しているグループが紐づいている作成済みのインデックス情報の一覧が返却されます。nameはインデックス名、document_countは登録済み文書数、accessIdsのgroupsがインデックスに紐づいているグループのID、usersがインデックスに紐づいているユーザのID、embeddingTypeがインデックスの文書登録時に使用される埋め込みモデルの種別(現在はopenaiのみ)、embeddingModelが埋め込みモデル名、コメントがインデックス作成時に指定した任意の文章が表示されます。

```
{"indexes":[
{"embeddingType":"openai",{"accessIds":{
"groups":["group_45b4f9ca-19d4-45d7-93eb-e469566f66dd",
"group_32377bc8-99d5-4137-872b-c6e97befd239"],"users":[
"user_fc79bb6a-b6d0-4fd3-9b45-a39525b15498",
"user_ad0942a4-8add-4110-882c-532b858fec1e"]},
"comment":"コメント","embeddingModel":"multilingual-e5-large",
"name":"test-index001","document_count":1},
{"embeddingType":"openai",{"accessIds":{"groups":[],
"users":["user_aa79bb6a-b6d0-4fd3-9b45-a39525b15498",
```

```
"user_bb0942a4-8add-4110-882c-532b858fec1e"]},
"comment":"test","embeddingModel":"multilingual-e5-large",
"name":"test-index002","document_count":10},
{"embeddingType":"openai",{"accessIds":{"groups":[
"group_45b4f9ca-19d4-45d7-93eb-e469566f66dd"],
"users":[]},"comment":"comment","embeddingModel":"multilingual-e5-large",
"name":"test-index003","document_count":5}]}
```

登録済みインデックス情報の変更を行う

以下のコマンド実行で登録済みのインデックス情報のうち、紐づいたグループ情報の削除とコメントの変更ができます。埋め込みモデルはインデックス作成時から変更することはできないのでご了承ください。実行時は指定するインデックスについて、事前にAPI USERが所属しているグループ、もしくはAPI USERがインデックスに紐づいていることをご確認ください。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。

```
curl -X PUT https://<サーバのドメイン名>/genai-search-api/index/updateIndex ¥
-H "Content-Type: application/json" -H "Authorization: Bearer <API Key>" ¥
--data "<リクエストパラメータ>"
```

以下はリクエストパラメータの例です。vectorIndexには情報を変更したいインデックス名を指定します。

以下は任意のパラメータになります。

accessIdsのgroupsとusersは現在のインデックスに紐づけられたグループ・ユーザ情報を削除したい場合のみ以下のように指定してください。変更がない場合は指定なし(`"accessIds":{}`)で実行してください。紐づけを削除した場合、管理ポータルからグループ情報やユーザ情報を紐づけしなおす必要がありますので、注意してください。

commentはインデックスに対する説明文などを指定してください。特に変更がない場合は指定なしにしてください。

```
{
  "vectorIndex": "test-index",
  "accessIds": {"groups":[], "users":[]},
  "comment" : "テスト用インデックス"
}
```

変更に成功すると以下のようにJson形式で指定したインデックスの名前が返却されます。

```
{"vectorIndex": "test-index"}
```

登録した文書情報を削除する

以下のコマンド実行で登録済みの文書情報をインデックスから削除できます。

実行時は指定するインデックスについて、事前にAPI USERが所属しているグループ、もしくはAPI USERがインデックスに紐づいていることをご確認ください。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。インデックス名には削除したい文書が登録されているインデックス名を指定してください。ファイル名には登録した文書のファイル名を指定して下さい。

```
curl -X DELETE ¥
https://<サーバのドメイン名>/genai-search-api/document/deleteDocument?
vectorIndex=<インデックス名>&filePath=<ファイル名> ¥
-H "Content-Type: application/json" -H "Authorization: Bearer <API Key>"
```

文書情報の削除に成功するとnull(何もエラーなどが表示されない)になります。

作成したインデックスを削除する

以下のコマンド実行で作成済みのインデックス情報を削除できます。

実行時は指定するインデックスについて、事前にAPI USERが所属しているグループ、もしくはAPI USERがインデックスに紐づいていることをご確認ください。

APIの実行方法

以下のcurlコマンドを用いて、APIを呼び出します。インデックス名には削除したいインデックス名を指定してください。

```
curl -X DELETE ¥
https://<サーバのドメイン名>/genai-search-api/index/deleteIndex?vectorIndex=<インデックス名> ¥
-H "Content-Type: application/json" -H "Authorization: Bearer <API Key>"
```

インデックスの削除に成功すると以下のように空のJsonが返ります。

```
{}
```