検索対話チュートリアル

はじめに

本書は、本サービスが提供する検索対話APIによって対話機能を実現するクライアントアプリケーションを作成するためのチュートリアルです。

本書の対象読者は以下を想定しています。

- 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

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。
サポートしているAPIとリクエスト・レスポンスの詳細については、APIリファレンスを参照してください。

- 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。

検索対話APIとは

LLMを使った対話機能を利用するAPIです。事前に登録された文書の内容に対して質問などを行いたい場合に使用します。

検索対話はサービス側で用意したシステムテンプレートを用いた対話と、システム管理者が作成したユーザテンプレートを用いた対話を行うことができます。

チュートリアルの流れ

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. 検索対話を利用する
3. (応用)複数のインデックスを対象にして検索対話を利用する
4. (応用) ユーザテンプレートによる検索対話を利用する
5. (応用) requestsライブラリを利用したpythonプログラムで検索対話を利用する

インデックスを作成する

本機能を利用するには事前に管理ポータルのインデックス一覧画面からインデックスを作成、検索したい文書を登録しておく必要があります。詳細は「管理ポータル操作ガイド（インデックス/文書管理編)」をご確認ください。

また文書登録時に任意のメタ情報を合わせて登録する、もしくはグループ認可の機能を使用したい場合は別途APIを使用する必要があります。詳細は「ベクトルDB管理API チュートリアル」をご確認ください。

検索対話を利用する

ここでは、システムテンプレートを用いた検索対話について説明します。

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の詳細は「Generative AI FW APIリファレンス」を参照してください。

```
curl -X POST https://<サーバのドメイン名>/genai-api/v1/searchchat -H "Content-Type: application/json" ¥
-d "<リクエストパラメータ>" -H "x-nec-genai-client-id:<ユーザId>" -H "Authorization: <API Key>"
```

以下はリクエストパラメータの例です。vectorIndexには作成したインデックス名を指定します。

```
{
  "userContent": "NECのcotomiについて教えてください",
  "vectorIndex": "test-index",
  "model": "cotomi-v3.0"
}
```

応答がJson形式で返却されます。sourceDocumentsには回答作成時に参照した文書情報が返却されます。

```
{"answer":"cotomi は...",sourceDocuments:[{"docContent":"cotomi 紹介資料...", ...}]}
```

複数のインデックスを対象にして検索対話を利用する

APIによって検索対話を実施する場合は複数のインデックスを指定することができます。
以下は複数のインデックスを対象に検索対話を行う際のリクエストパラメータの例です。
vectorIndexesのパラメータを使用することでインデックスを複数指定することができます。

```
{
  "userContent": "NECのcotomiについて教えてください",
  "vectorIndexes": ["test-index1","test-index2","test-index3"],
  "model": "cotomi-v3.0"
}
```

応答がJson形式で返却されます。sourceDocumentsには回答作成時に複数のインデックスにまたがって参照された文書情報が返却されます。

```
{"answer":"cotomi は...",sourceDocuments:[{"docContent":"cotomi 紹介資料...", ...}]}
```

ユーザテンプレートによる検索対話を利用する

システム管理者が作成した検索対話用のユーザテンプレートを用いて対話することもできます。

テンプレートを作成する時は、システムテンプレート（検索対話）を引用して作成することを推奨します。

システムテンプレートに埋め込まれているパラメータ({userContent}や{context})を削除すると機能しなくなるため、削除しないようにしてください。パラメータの説明や操作の詳細は「管理ポータル操作ガイド（テンプレート機能編)」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。APIの詳細は「Generative AI FW APIリファレンス」を参照してください。

```
curl -X POST https://<サーバのドメイン名>/genai-api/v1/searchchat -H "Content-Type: application/json" ¥
-d "<リクエストパラメータ>" -H "x-nec-genai-client-id:<ユーザId>" -H "Authorization: <API Key>"
```

以下はリクエストパラメータの例です。templateIdにはユーザテンプレートのIDを指定します。

```
{
  "templateId" : "template_12345678-90ab-cdef-ghij-zzz",
  "userContent": "NECのcotomiについて教えてください",
  "vectorIndex": "test-index",
  "model": "cotomi-v3.0"
}
```

応答がJson形式で返却されます。

```
{"answer":"cotomi は...",sourceDocuments:[{"docContent":"cotomi 紹介資料...", ...}]}
```

requestsライブラリを利用したpythonプログラムで検索対話を利用する

requestsライブラリのインストール

インストールコマンドの例

```
pip install requests
```

Pythonコードを実装する

Pythonコードの例

Pythonのエディタを開き、下記のコードを記述します。ファイル名を「search_main.py」として保存します。

```
import requests

# KEYはAPIキー
KEY = 'abcdefg1234567890'
# BASEはサービスのURL
BASE = 'https://<サーバのドメイン名>/genai-api/v1'
# MODELはエイリアス名
MODEL = 'cotomi-v3.0'
# 作成したインデックス名
INDEX = 'test-index'

def func():
    url = BASE + '/searchchat'
    key = 'Bearer ' + KEY
    payload = { "userContent": "NECのcotomiについて教えてください",
                "vectorIndex": INDEX,
                "model": MODEL }
    headers = { 'content-type': 'application/json',
                'x-nec-genai-client-id': 'ABCDEF',
                'Authorization': key }
    # 非ストリーム形式
    response = requests.post(url, json=payload, headers=headers)
    print(response.status_code)
    print(response.text)
    #print(response.headers)

if __name__ == '__main__':
    func()

```

Pythonコードを利用した場合においても「vectorIndex」のかわりに、「vectorIndexes」のパラメータを指定することで複数のインデックスにまたがった検索対話を行うことができます。

```
   payload = { "userContent": "NECのcotomiについて教えてください",
               "vectorIndexes": ["test-index1","test-index2","test-index3"],
               "model": MODEL }
```

Pythonの実行

実行コマンド

```
python search_main.py
```

実行結果の例

```
200
{"answer":"cotomi は...",sourceDocuments:[{"docContent":"cotomi 紹介資料...", ...}]}
```