日日是Oracle APEX

Oracle APEXを使った作業をしていて、気の付いたところを忘れないようにメモをとります。

2023年7月7日金曜日

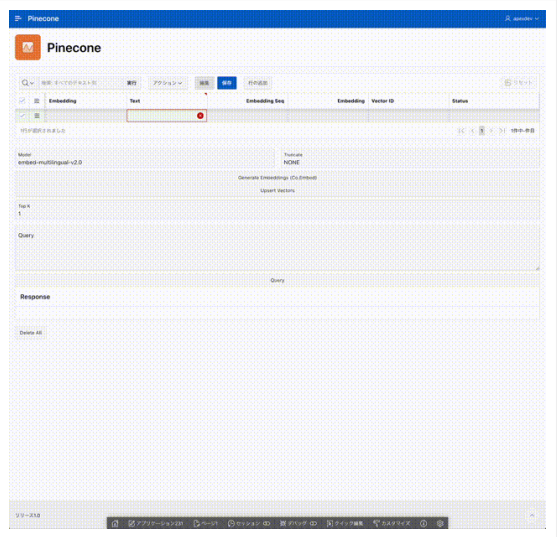
Cohereでembeddingを生成しPineconeに保存して問い合わせてみる

ベクトル・データベースのPineconeは、無料でインデックスをひとつ作成することができます。その無料枠で作成したインデックスにCohereのEmbed APIで生成したembeddingを保存し、問い合わせを実行するアプリケーションを作ってみました。

以下の処理を実装しています。

1. オラクル・データベースの表に検索対象となる文書を保存します。
2. 上記の保存された文書を引数にしてCohere Embed APIを呼び出し、embeddingを生成します。生成したembeddingは、文書に紐づけてオラクル・データベースに保存します。
3. オラクル・データベースに保存されている文書のidとembeddingを、Pineconeのインデックスにupsertします。
4. 画面から問い合わせとなる文字列を入力します。問い合わせ文字列を引数にしてCohere Embed APIを呼び出し、embeddingを生成します。生成したembeddingでPineconeのインデックスにQueryを発行します。
5. Pineconeより類似した文書のIDが返されるので、オラクル・データベースに保存された文書をIDを指定して取り出し、画面に印刷します。

以下のような動作になります。



上記のアプリケーションのエクスポートは以下に置いてあります。
https://github.com/ujnak/apexapps/blob/master/exports/pinecone-and-cohere-sample.zip

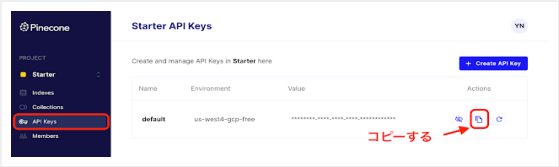
上記のアプリケーションを一からつくるということもないので、実装のポイントのみを解説します。

Pineconeのアカウントの作成方法については、他の説明を参照してください。 Google、GitHubまたはMicrosoftのアカウントと連携するため、それらのアカウントがあればすぐに作成できます。また、有料の機能を使用するにはアカウントをアップグレードする必要があります。

Pineconeの準備作業

PineconeのAPIを呼び出す際に使用する**APIキー**を確認します。このAPIキーより、Oracle APEXのワークスペースに**Web資格証明**を作成します。

Pineconeのコンソールを開き、ナビゲーション・メニューより**API Keys**を開きます。作成済みのAPIキーを**コピー**します。

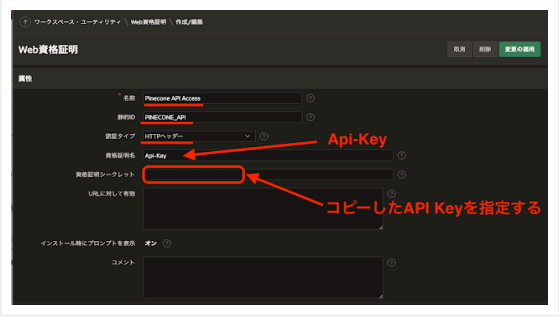


ワークスペース・ユーティリティより**Web資格証明**を開き、PineconeのAPI呼び出し時に指定するWeb資格証明を作成します。

Web資格証明の**名前**は**Pinecone API Access**、**静的ID**は**PINECONE_API**としています。**静的ID**は、これから紹介するPL/SQLコードに含まれています。

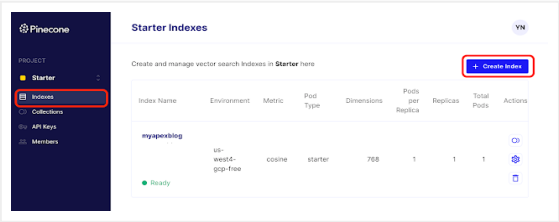
資格証明タイプとして**HTTPヘッダー**を選び、**資格証明名**として**Api-Key**を指定します。**資格証明シークレット**として、Pineconeのコンソールでコピーした**API Key**を設定します。

PineconeのAPIはHTTPヘッダー**Api-Key**の値として、APIキーを渡すことで認証されます。



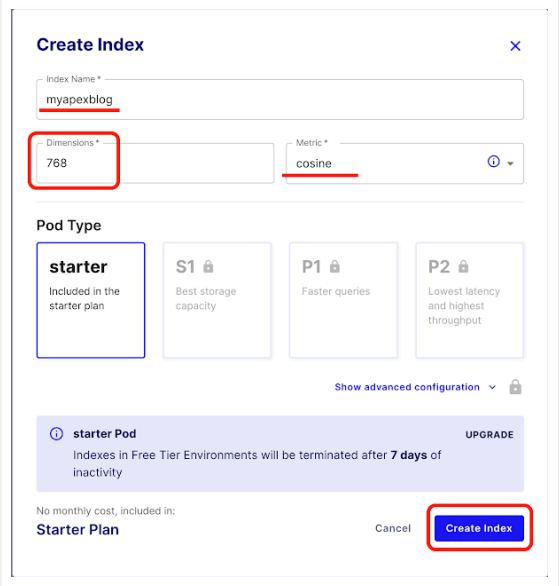
Pineconeのインデックスを作成します。

Pinecone Consoleの**Indexes**を開き、**Create Index**を実行します。

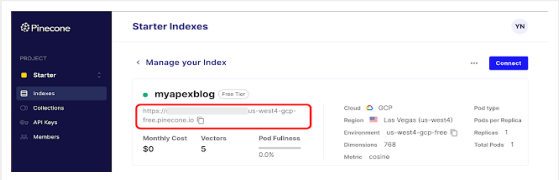


作成するインデックスの**Dimensions**は**768**を指定します。今回はCohereのEmbed APIを呼び出してembeddingを生成します。その際に（日本語を扱うため）**model**として**embed-multilingual-v2.0**を指定します。このembed-multilingual-v2.0のサイズが768なので、Pineconeのインデックスの Dimensionsを768にします。CohereのモデルのEmbedding SizeはCo.EmbedのAPIのリファレンスに記載されています。

Metricとして**cosine**（コサイン類似度）を選択しています。こちらは他に**dotproduct**（ドット積）、**euclidean**（ユークリッド距離）を選択することができます。



作成したインデックスのURLを確認し、コピーします。



このインデックスはAPEXアプリケーションの**アプリケーション定義**の**置換**に、置換文字列**G_INDEX**の**置換値**として設定します。

アプリケーションのコード中ではG_INDEXとして、PineconeのAPIのエンドポイントを参照します。

Cohereの準備については、こちらの記事を参照してください。

以上でPineconeを使用する準備ができました。

表VEC_EMBEDDINGSの作成

以下のDDLを実行し、表**VEC_EMBEDDINGS**を作成します。

```
  -- create tables
create table vec_embeddings (
    id                              number generated by default on null as identity
                                    constraint vec_embeddings_id_pk primary key,
    text                            varchar2(512 char) not null,
    embedding_id                    varchar2(40 char),
    embedding_seq                   number,
    embedding                       clob check (embedding is json),
    vector_id                       varchar2(40 char),
    status                          varchar2(1 char) default on null 'C' constraint vec_embeddin
                                    check (status in ('C','U')),
    created                         date not null,
    created_by                      varchar2(255 char) not null,
    updated                         date not null,
    updated_by                      varchar2(255 char) not null
)
;

-- comments
comment on column vec_embeddings.embedding is 'embedding genereted by Cohere, json array of flo
comment on column vec_embeddings.embedding_id is 'id in each Cohere Co.Embed response';
comment on column vec_embeddings.embedding_seq is 'identify embedding in Co.Embed response';
comment on column vec_embeddings.status is 'C for Created, not yet stored in Pinecone, U stored
comment on column vec_embeddings.vector_id is 'object id of embedding/vector stored in Pincone,


-- triggers
create or replace trigger vec_embeddings_biu
    before insert or update
    on vec_embeddings
    for each row
begin
    if inserting then
        :new.created := sysdate;
        :new.created_by := coalesce(sys_context('APEX$SESSION','APP_USER'),user);
    end if;
    :new.updated := sysdate;
```

```
    :new.updated_by := coalesce(sys_context('APEX$SESSION','APP_USER'),user);
end vec_embeddings_biu;
/

-- load data
```

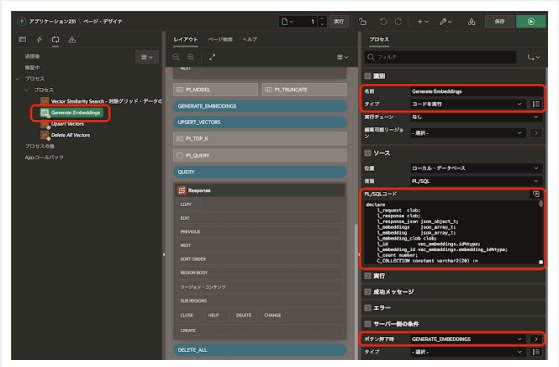
vec_emgeddings.sql hosted with ❤️ by GitHub view raw

アプリケーションのユーザーが入力するのは、列**TEXT**の値だけです。

Cohereによるembeddingの生成

ボタン**GENERATE_EMBEDDINGS**をクリックしたときに、列TEXTの値をまとめてCohere Embed APIを呼び出し、embeddingを生成します。

ボタンのクリックにより実行されるプロセスは、**Generate Embeddings**として作成されています。



ソースの**PL/SQLコード**は以下です。

```
declare
    l_request  clob;
    l_response clob;
    l_response_json json_object_t;
    l_embeddings    json_array_t;
    l_embedding     json_array_t;
    l_embedding_clob clob;
    l_id            vec_embeddings.id%type;
    l_embedding_id vec_embeddings.embedding_id%type;
    l_count number;
    C_COLLECTION constant varchar2(20) := 'COHERE_EMBED_REQUEST';
    e_embed_exception exception;
begin
    /*
     * embeddingの生成が必要なデータを確認する。
     */
```

```
select count(*) into l_count from vec_embeddings where embedding is null;
if l_count = 0 then
    /* embeddingが未生成のデータがないため終了。 */
    return;
end if;
/*
 * 表VEC_EMBEDDINGSから、embeddingの生成対象であるデータをAPEXコレクションにコピーする。
 */
apex_collection.create_or_truncate_collection(C_COLLECTION);
for r in (
    select id, text from vec_embeddings where embedding is null order by id
)
loop
    apex_collection.add_member(
        p_collection_name => C_COLLECTION
        ,p_n001 => r.id
        ,p_c001 => r.text
    );
end loop;
/* 念の為、シーケンスIDが1から連番になるようにする。 */
apex_collection.resequence_collection(C_COLLECTION);
/*
 * APEXコレクションからCo.Embedのリクエストを生成する。
 */
select json_object(
    key 'texts' value
    (
        select json_arrayagg(c001 order by seq_id asc) from apex_collections
        where collection_name = C_COLLECTION
    )
    -- モデルを指定する。
    ,key 'model'    value :P1_MODEL
    -- トランケートを指定する。
    ,key 'truncate' value :P1_TRUNCATE
returning clob)
into l_request
from dual;
apex_debug.info(l_request);
/*
 * Cohere Co.Embedを呼び出す。
 * https://docs.cohere.com/reference/embed
 */
apex_web_service.clear_request_headers;
apex_web_service.set_request_headers('Accept','application/json', p_reset => false);
apex_web_service.set_request_headers('Content-Type','application/json', p_reset => false);
l_response := apex_web_service.make_rest_request(
    p_url => 'https://api.cohere.ai/v1/embed'
```

```
        ,p_http_method => 'POST'
        ,p_body => l_request
        ,p_credential_static_id => 'COHERE_API'
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info(l_response);
        raise e_embed_exception;
    end if;
    /*
     * 生成されたembeddingを取り出し、表VEC_EMBEDDINGSに保存する。
     */
    l_response_json := json_object_t.parse(l_response);
    l_embedding_id  := l_response_json.get_string('id');
    l_embeddings    := l_response_json.get_array('embeddings');
    for i in 1..l_embeddings.get_size
    loop
         l_embedding := json_array_t(l_embeddings.get(i-1));
         l_embedding_clob := l_embedding.to_clob; /* embedding自体はFLOATの配列として、JSONのまま保存す
         select n001 into l_id from apex_collections where collection_name = C_COLLECTION and s
         update vec_embeddings set
            embedding_id = l_embedding_id
            ,embedding_seq = i
            ,embedding = l_embedding_clob
         where id = l_id;
     end loop;
end;
```

generate_embeddings_calling_cohere.sql hosted with ❤️ by GitHub view raw

列**EMBEDDING**が未設定の行をAPEXコレクションにコピーし、それを元にEmbed APIのリクエストを作成しています。Embed APIのリクエストには複数のテキストを含めることができます。

APIのリファレンスによると、１つのリクエストに含めることができるテキストの数は96が上限で、それぞれのテキストのトークンは512を超えないことが推奨されています。

要確認なのですが、英語のモデルであればトークン数は単語数と同じと思うのですが、多言語のモデルembed-multilingual-v2.0で特に日本語の場合は、トークン数は文字数と同じではないかと思います。それもあって表VEC_EMBEDDINGSの列TEXTをVARCHAR2(512 CHAR)としています。

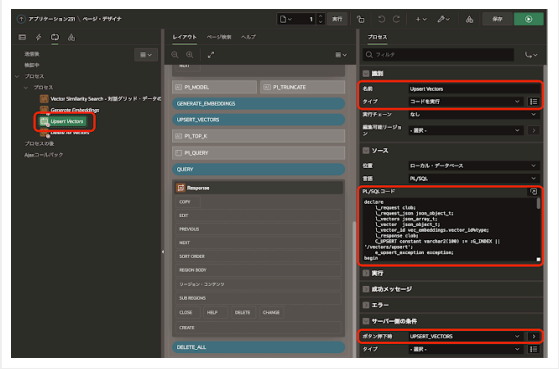
CohereのEmbed APIのレスポンスにid属性が含まれています。ただし、リクエストに複数のテキストが含まれ、それぞれのembeddingがレスポンスに含まれています。idだけではレスポンスに含まれる個々のテキストとembeddingの組み合わせを特定できないため、テキスト（およびembedding）の出現順序を列EMBEDDING_SEQとして保存しています。列EMBEDDING_ID（Cohere Embed APIのレスポンスのid）と組み合わせて、テキストとembeddingを一意に特定します。

この列EMBEDDING_IDとEMBEDDING_SEQは、PineconeのUpsertを呼び出す際のベクトルのIDになります。

PineconeへのUpsertの実行

ボタン**UPSERT_VECTORS**をクリックしたときに、列EMBEDDINGにデータが記載され、列STATUSがCである行をまとめて、PineconeのUpsertリクエストを発行します。

ボタンのクリックにより実行されるプロセスは、**Upsert Vectors**として作成されています。



ソースの**PL/SQLコード**は以下です。

```
declare
    l_request clob;
    l_request_json json_object_t;
    l_vectors json_array_t;
    l_vector  json_object_t;
    l_vector_id vec_embeddings.vector_id%type;
    l_response clob;
    C_UPSERT constant varchar2(100) := :G_INDEX || '/vectors/upsert';
    e_upsert_exception exception;
begin
    /*
     * Cohereで生成したそれぞれのembeddingが、Pineconeでのvectorにあたる。
     * vectorオブジェクト(idとvalues = embeddingを含むオブジェクト)の配列が、
     * PineconeのインデックスへのUpsertリクエストになる。
     */
    l_vectors := json_array_t();
    for r in (
        select embedding_id, embedding_seq, embedding from vec_embeddings
        where status = 'C' and embedding is not null
    )
    loop
       l_vector_id := r.embedding_id || '-' || r.embedding_seq;
       l_vector := json_object_t();
       l_vector.put('id', l_vector_id);
       l_vector.put('values', json_array_t(r.embedding));
       l_vectors.append(l_vector);
       update vec_embeddings set vector_id = l_vector_id, status = 'U'
       where embedding_id = r.embedding_id and embedding_seq = r.embedding_seq;
```

```
    end loop;
    l_request_json := json_object_t();
    l_request_json.put('vectors', l_vectors);
    l_request := l_request_json.to_clob;
    /*
     * PineconeのUpsertを呼び出す。
     */
    apex_web_service.clear_request_headers;
    apex_web_service.set_request_headers('Accept','application/json',p_reset => false);
    apex_web_service.set_request_headers('Content-Type','application/json',p_reset => false);
    l_response := apex_web_service.make_rest_request(
        p_url => C_UPSERT
        ,p_http_method => 'POST'
        ,p_body => l_request
        ,p_credential_static_id => 'PINECONE_API'
    );
    /* レスポンス本文は解析不要。 */
    apex_debug.info(l_response);
    if apex_web_service.g_status_code <> 200 then
        raise e_upsert_exception;
    end if;
end;
```

pinecone_upsert_vectors.sql hosted with ❤️ by GitHub view raw

列EMBEDDING_IDとEMBEDDING_SEQよりidの値を作り、valuesにはCohereのEmbed APIが返してきたembeddingをそのまま渡します。vectors属性に、このオブジェクトの配列を指定し、PineconeのUpsertのリクエストとしています。

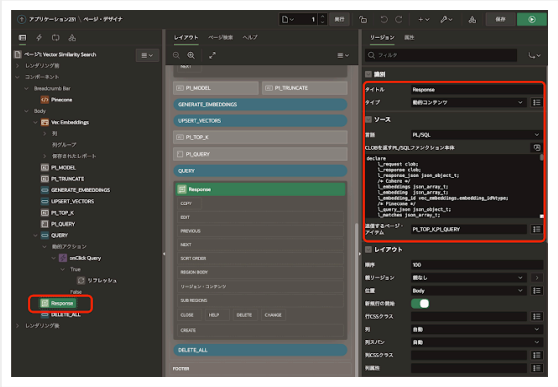
ここで指定したidの値を、表VEC_EMBEDDINGSの列VECTOR_IDとして保存します。

Pineconeのインデックスに対する問い合わせは、このidの値を検索結果として返します。表VEC_EMBEDDINGSの列VECTOR_IDから、元のテキストを見つけることができます。

Pineconeのインデックスへの問い合わせ

ボタン**QUERY**をクリックすると、Pineconeのインデックスへの問い合わせが発行されます。ボタン自体は、動的アクションによって動的コンテンツのリージョンである**Response**をリフレッシュしています。

PineconeのインデックスへのQuery APIの発行および結果の表示は、動的コンテンツに実装されています。



ソースの**CLOBを返すPL/SQLファンクション本体**は以下です。

```
declare
    l_request clob;
    l_response clob;
    l_response_json json_object_t;
    /* Cohere */
    l_embeddings json_array_t;
    l_embedding  json_array_t;
    l_embedding_id vec_embeddings.embedding_id%type;
    /* Pinecone */
    l_query_json json_object_t;
    l_matches json_array_t;
    l_vector  json_object_t;
    l_vector_id vec_embeddings.vector_id%type;
    l_score   number;
    l_text    vec_embeddings.text%type;
    C_QUERY constant varchar2(80) := :G_INDEX || '/query';
    e_query_exception exception;
    l_output clob;
begin
    /*
     * 問合せがない時はなにもしない。
     */
    if :P1_QUERY is null then
        return '';
    end if;
    /*
     * 最初にCo.Embedを呼び出して、Queryのembeddingを生成する。
     */
    select json_object(
        key 'texts' value json_array(:P1_QUERY)
        -- モデルを指定する。
        ,key 'model'    value :P1_MODEL
        -- トランケートを指定する。
        ,key 'truncate' value :P1_TRUNCATE
    returning clob)
```

```
into l_request
from dual;
/*
 * Cohere Co.Embedを呼び出す。
 * https://docs.cohere.com/reference/embed
 */
apex_web_service.clear_request_headers;
apex_web_service.set_request_headers('Accept','application/json', p_reset => false);
apex_web_service.set_request_headers('Content-Type','application/json', p_reset => false);
l_response := apex_web_service.make_rest_request(
    p_url => 'https://api.cohere.ai/v1/embed'
    ,p_http_method => 'POST'
    ,p_body => l_request
    ,p_credential_static_id => 'COHERE_API'
);
-- apex_debug.info(l_response);
l_response_json := json_object_t.parse(l_response);
l_embedding_id  := l_response_json.get_string('id');
l_embeddings    := l_response_json.get_array('embeddings');
l_embedding     := json_array_t(l_embeddings.get(0));
/*
 * Pineconeに問い合わせる。
 */
l_query_json := json_object_t();
l_query_json.put('includeValues', false);
l_query_json.put('includeMetadata', false);
l_query_json.put('vector', l_embedding);
l_query_json.put('topK', :P1_TOP_K);
l_request := l_query_json.to_clob;
/* queryを呼び出す。 */
apex_web_service.clear_request_headers;
apex_web_service.set_request_headers('Accept','application/json',p_reset => false);
apex_web_service.set_request_headers('Content-Type','application/json',p_reset => false);
l_response := apex_web_service.make_rest_request(
    p_url => C_QUERY
    ,p_http_method => 'POST'
    ,p_body => l_request
    ,p_credential_static_id => 'PINECONE_API'
);
-- apex_debug.info(l_response);
if apex_web_service.g_status_code <> 200 then
    apex_debug.info(l_response);
    raise e_query_exception;
end if;
/*
 * レスポンスを動的コンテンツとして印刷する。
 */
```

```
    l_response_json := json_object_t.parse(l_response);
    l_matches := l_response_json.get_array('matches');
    for i in 1..l_matches.get_size
    loop
        l_vector := json_object_t(l_matches.get(i-1));
        l_vector_id := l_vector.get_string('id');
        l_score     := l_vector.get_number('score');
        select text into l_text from vec_embeddings where vector_id = l_vector_id;
        l_output := l_output || '<p>id = ' || l_vector_id || ', score = ' || l_score || '</p>';
        l_output := l_output || '<p>' || l_text || '</p>';
    end loop;
    return l_output;
end;
```

query_pinecone.sql hosted with ❤️ by GitHub view raw

問い合わせ文字列を引数として、CohereのEmbed APIを呼び出しembeddingを生成しています。

そのembeddingを、PineconeのQuery APIのリクエストのvectorとして渡しています。

結果としてインデックスに保存されているベクトルのidとscoreが返されます。idに一致するテキストを、表VEC_EMBEDDINGSの列VECTOR_IDから検索し表示しています。

今回作成したアプリケーションの説明は以上になります。

ベクトル類似性検索を試してみたい、ということでCohereのEmbed APIとPineconeを使ってみました。とりあえず動かすことはできたので目的は達成したのですが、インデックスのメンテナンスや検索結果の評価方法など、色々と考えないといけないことは多そうです。

Oracle APEXのアプリケーション作成の参考になれば幸いです。

完

Yuji N. 時刻: 18:37

共有

自己紹介

Yuji N.

日本オラクル株式会社に勤務していて、Oracle APEXのGroundbreaker Advocateを拝命しました。
こちらの記事につきましては、免責事項の参照をお願いいたします。

詳細プロフィールを表示