

日々是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25日 火曜日

Llama.cpp_serverのOpenAI互換APIを呼び出してIn-context Learningを行なう

先日、Cohereをベクトル埋め込みと応答の生成に、Pineconeをベクトル類似度検索に使用したIn-context Learningを行なうAPEXアプリケーションを作成してみました。

このアプリケーションのCohereを使った処理を、Llama.cppのOpenAI互換APIの呼び出しに変えてみます。Llama.cppでOpenAI互換APIを使用する方法については、こちらの記事で紹介しています。

Llama2の7Bのモデルで生成されるベクトル埋め込みの次元は4096、13Bのモデルでは5120になります。Pineconeのインデックスは使用するモデルの次元数に合わせて作り直す必要があります。

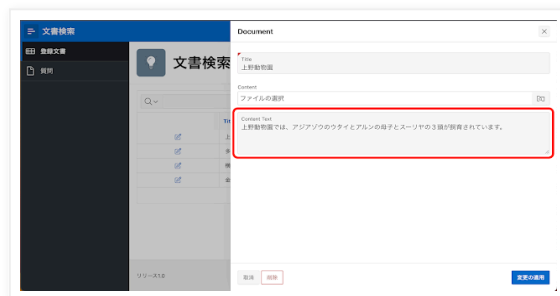
改変したAPEXアプリケーションのエクスポートは以下です。

<https://github.com/ujnak/apexapps/blob/master/exports/vector-documents-search-openai.zip>

知識として以下の4行を登録します。

「上野動物園では、アジアゾウのウタイとアルンの母子とスーリヤの3頭が飼育されています。」
「多摩動物公園ではアヌーラ、アマラ、ビイドゥラの3頭のスリランカゾウとアフリカゾウのトムの1頭が飼育されています。」
「横浜動物園ズーラシアではラスクマルとシュリーの2頭のアジアゾウが飼育されています。」
「金沢動物園ではボンとヨーコの2頭のインドゾウが飼育されています。」

アプリケーションはファイルのアップロードだけではなく、知識を直接フォームから登録できるように改変しています。



アップロードするファイルもPDFではなく、セパレータとして'---'または'=== 'を含めたプレーンテキストの文書を受け付けるように変更しています。

質問を実行してみます。

質問：「金沢動物園で飼育されているゾウの種類はなんですか。」

回答：「おはようございます！金沢動物園で飼育されているのは、インドゾウです。」

APIのリクエストに含める**messages**として、**role**が**system**である**content**と**role**が**user**である**content**を含めるようにしています。

roleが**system**の**content**として、ページ・アイテム**P3_PROMPT_SYSTEM**に入力した文章をそのまま使います。**role**が**user**の**content**として、ページ・アイテム**P3_PROMPT_TEMPLATE**に記述されたテンプレートの**{context_str}**の部分をインデックスの検索結果として得られた知識（チャンク）で置き換え、そして、**{query_str}**の部分を質問で置き換えた文章を使います。

ページ・アイテムP3_PROMPT_SYSTEMは、以下をデフォルトとして設定しています。

あなたは日本語を話すアシスタントです。

ページ・アイテムP3_PROMPT_TEMPLATEは、以下をデフォルトとして設定しています。

以下の情報が与えられています。

```
{context_str}
```

この情報から、次の質問に教えてください。: {query_str}

APEXのアプリケーションから質問を行うと、Llama.cppからは以下のログが出力されます。

```
llama_print_timings:      load time = 24694.52 ms
llama_print_timings:      sample time =      0.00 ms /      1 runs  (      0.00 ms per
token,      inf tokens per second)
llama_print_timings: prompt eval time = 11806.55 ms /    34 tokens (   347.25 ms per
token,      2.88 tokens per second)
llama_print_timings:      eval time =      0.00 ms /      1 runs  (      0.00 ms per
token,      inf tokens per second)
llama_print_timings:      total time = 11895.69 ms
INFO:      ::1:39706 - "POST /v1/embeddings HTTP/1.0" 200 OK
Llama.generate: prefix-match hit
```

```
llama_print_timings:      load time = 24694.52 ms
llama_print_timings:      sample time =  40.99 ms /    41 runs   (    1.00 ms per
token, 1000.22 tokens per second)
llama_print_timings: prompt eval time = 55983.91 ms /   162 tokens (   345.58 ms per
token,  2.89 tokens per second)
```

```
llama_print_timings:          eval time = 15736.55 ms /    40 runs   (   393.41 ms per token,
                               2.54 tokens per second)
llama_print_timings:          total time = 72299.14 ms
INFO:      150.136.133.92:0 - "POST /v1/chat/completions HTTP/1.0" 200 OK
```

質問文のベクトル埋め込みの計算(POST /v1/embeddings)で、約12秒、回答の生成(POST /v1/chat/completions)に72秒程度の時間がかかっています。非常に簡単な質問でこの程度の処理時間なので、長文はほぼ扱えません。

次の質問です。

質問：「金沢動物園で飼育されているアジアゾウの名前を教えてください。」

回答：「はい、すみません。金沢動物園で飼育されているアジアゾウの名前は「ボン」と「ヨーコ」です。」

正解に見えます。

次の質問です。

質問：「多摩動物公園で飼育されているアジアゾウの名前を教えてください。」

回答：「はい、あなたに答えます。多摩動物公園で飼育されているアジアゾウの名前は、アヌーラです。」

多摩動物公園にはアヌーラの他にアマラとビィドゥラが飼育されていますが、必ずしも不正解とは言えないでしょう。

質問を変えてみます。「すべて」を追加しました。

質問：「多摩動物公園で飼育されているアジアゾウの名前をすべて教えてください。」

回答：「はい、あなたに答えます。以下のように、多摩動物公園で飼育されているアジアゾウの名前は、アヌーラ、アマラ、ビィドゥラ、トムです。」

回答にアフリカゾウのトムが含まれているため、不正解です。

以下の質問をしてみます。知識は不要なので、テンプレートを変更し{query_str}のみを記載します。

質問：「インドゾウとスリランカゾウとアフリカゾウのうち、アジアゾウに含まれるゾウはどれですか？」

回答：「アジアゾウに含まれるゾウは、インドゾウとスリランカゾウの2種類です。」

正解です。知識としてはあるのかもしれませんが。質問のテンプレートを以下に変更し、同じ質問をしてみます。

以下の情報が与えられています。

{context_str}

アジアゾウに含まれるゾウは、インドゾウとスリランカゾウの2種類です。アジアゾウにアフリカゾウは含まれません。

この情報から、次の質問に日本語で答えてください。: {query_str}

回答：「はい、あなたに答えます。以下が多摩動物公園で飼育されているアジアゾウの名前です。

* アヌーラ

* アマラ

* ビイドゥラ

私たちは、スリランカゾウとアフリカゾウを飼育していません。」

トムは除かれましたが、最後に正しくない回答が加わっています。

以下の質問は回答が間違っています。

質問：「アジアゾウのアルンが飼育されている動物園を教えてください。」

回答：「あなたは、多摩動物公園でアジアゾウのアルンが飼育されています。」

Pineconeのインデックスを検索したときにTop Kとして1を与えていて、以下の知識だけがプロンプトに含まれました。

「多摩動物公園ではアヌーラ、アマラ、ビイドゥラの3頭のスリランカゾウとアフリカゾウのトムの1頭が飼育されています。」

まったく知識がない状態なので、適当な回答が生成されています。

Top Kを2にし、2番目に類似度が高い知識もプロンプトに含めてみました。2番目の知識に上野動物園の情報が含まれたので、正解に見える回答が得られています。

回答：「はい、わかりました。アジアゾウのアルンは上野動物園で飼育されています。」

少し質問を変えてみます。Top Kに2を指定します。

質問：「アルンのお母さんのゾウの名前を教えてください。」

回答：「はい、わかりました。アルンのお母さんのゾウの名前はスーリヤです。」

間違っています。知識として与えている文章が認識できてないようです。知識の方を以下の文章に変更します。

「上野動物園では、アジアゾウのアルンとその母のウタイ、その他にスーリヤの3頭が飼育されています。」

同じ質問をします。

回答：「はい、わかりました。アルンのお母さんのゾウの名前は「ウタイ」です。」

正解です。

実用できるかどうかの判断は、OpenAIのAPIやLlama2でも70Bのモデルを使って行なう必要があると思いますが、なかなか一筋縄ではいかないように感じます。

以下より、以前のアプリケーションからの変更点を説明します。

表定義ですが、表KB_CHUNKSの列SPLIT_CHARをSPLIT_CHARSに変更し、VARCHAR2(1)からVARCHAR2(8)に変更しています。その他、長い文字列を保存する列の型をVARCHAR2(4000)からCLOBに変更しています。

```
# prefix: kb
documents
  title          vc160 /nn
  content         file
  content_text   clob
  is_failed      vc1

chunks
  document_id num /fk documents
  seq         num
  chunk       clob
  embedding   json
  split_chars vc8
  is_indexed  vc1

questions
  question      vc4000
  embedding     json
  asked_by      vc80
  asked_date    date /default sysdate

answers
  question_id num /fk questions
  chunk_id    num /fk chunks
  score       num
  answered_date date /default sysdate

responses
  question_id num /fk questions
  iteration   num
  prompt      clob
  generated_answer clob
```

kb_openai.txt hosted with ❤️ by GitHub

[view raw](#)

OpenAI互換のAPIを呼び出すように改変したパッケージKB_LLM_UTILのコードは以下になります。

```
create or replace package kb_llm_util as
/**
 * OpenAIのembeddingとchat/completions APIを呼び出すように
 * 改変したパッケージ。
 *
 * Llama_cpp.serverのOpenAI互換APIを使用しているので、本家の
 * OpenAIのAPIで変更が必要などところもある可能性があります。
```

```

*/

/**
 * 表KB_DOCUMENTSのBLOB列CONTENTをCLOB列のCONTENT_TEXTに
 * 単純にコピーする。
 *
 * PDFをソースにするといったことはせず、MIMEタイプはtext/plainを想定
 * している。
 */
procedure apply_auto_filter(
    p_id in number
);

/**
 * 表KB_DOCUMENTSの列CONTENT_TEXTに保存されている文章をチャンクに分割する。
 * 分割されたチャンクは表KB_CHUNKSに保存される。
 *
 * LlamaIndexでいうところのNode Parserに該当する処理を行う。
 * https://gpt-index.readthedocs.io/en/latest/core\_modules/data\_modules/node\_parsers/root.html
 *
 * どのような形でチャンクに分割するかはとても重要。以下では、
 * CHR(10) || '---' || CHR(10) または CHR(10) || '=== ' || CHR(10) で
 * チャンクに分割する。
 */
procedure split_into_chunks(
    p_id in number
    ,p_primary_separator in varchar2
    ,p_secondary_separator in varchar2
    ,p_limit in number default 4000
);

/**
 * 表KB_CHUNKSに保存した列CHUNKのベクトル埋め込みを作成する。
 * OpenAIの/v1/embeddingを呼び出す。
 * 実際はLlama_cpp.serverで、Llama2の7bは4096、13Bは5120の
 * 次元のベクトル埋め込みを生成する。
 *
 * 生成したベクトル埋め込みは列EMBEDDINGに保存する。
 */
procedure generate_embeddings(
    p_id in number
    ,p_collection_name in varchar2 default 'EMBEDDINGS'
    ,p_model_name in varchar2 default 'text-embedding-ada-002'
    ,p_endpoint in varchar2 default null
    ,p_cred_id in varchar2 default null
);

```

```

/**
 * 生成したベクトル埋め込みをPineconeのインデックスにUpsertする。
 */
procedure upsert_vectors(
    p_id in number
    ,p_index in varchar2
    ,p_pinecone_cred in varchar2 default 'PINECONE_API'
);

/**
 * 質問の送信と回答の表示。
 *
 * 質問の文字列のベクトル埋め込みを生成し、Pineconeのインデックスを検索する。
 * 回答数はp_top_kで指定する。検索結果のチャンクは、scoreの良い順番で連結する。
 *
 * LlamaIndexでいうところのResponse Synthesizerに該当する処理を行う。
 * https://gpt-index.readthedocs.io/en/latest/core\_modules/query\_modules/response\_synthesizers/
 * OpenAIのAPIであれば、LlamaIndexでのpromptは以下。
 * https://gpt-index.readthedocs.io/en/latest/core\_modules/model\_modules/prompts.html
 *
 * プロンプトの生成方法については、暫定的なもので要調整。
 */
procedure ask(
    p_question in varchar2
    ,p_prompt_system in varchar2
    ,p_top_k in number
    ,p_index in varchar2
    ,p_answer out varchar2
    ,p_question_id out number
    ,p_score_limit in number default 0
    ,p_model_name in varchar2 default 'text-embedding-ada-002'
    ,p_endpoint in varchar2 default null
    ,p_cred_id in varchar2 default null
    ,p_pinecone_cred in varchar2 default 'PINECONE_API'
    ,p_generate_model_name in varchar2 default 'gpt-3.5-turbo'
    ,p_temperature in number default 0.9
    ,p_max_tokens in number default 256
    ,p_prompt_template in clob
);

/**
 * 削除された文書のベクトルをPineconeのインデックスから削除する。
 */
procedure delete_vectors(
    p_id in number
    ,p_index in varchar2
    ,p_pinecone_cred in varchar2 default 'PINECONE_API'

```

```

);
end;
/

create or replace package body kb_llm_util as
C_OPENAI_API_TIMEOUT constant number := 360; -- 6 min.

/**
 * Extract text string from BLOB column.
 */
procedure apply_auto_filter(
    p_id in number
)
as
    l_content      kb_documents.content%type;
    l_content_text kb_documents.content_text%type;
    l_is_failed     kb_documents.is_failed%type := 'N';
begin
    /* assume mime type is 'text/plain' so simply convert blob to clob. */
    update kb_documents set content_text = to_clob(content), is_failed = 'N'
    where id = p_id;
end apply_auto_filter;

/**
 * Split document into chunks.
 */
procedure split_into_chunks(
    p_id in number
    ,p_primary_separator in varchar2
    ,p_secondary_separator in varchar2
    ,p_limit in number
)
as
    l_content_text kb_documents.content_text%type;
    l_chunk        kb_chunks.chunk%type;
    l_seq          kb_chunks.seq%type;
    l_split_chars  kb_chunks.split_chars%type;
    l_split        number;
begin
    /* Delete chunks currently exists for update. */
    delete from kb_chunks where document_id = p_id;
    select content_text into l_content_text from kb_documents where id = p_id;

    l_seq := 1;
    while true
    loop
        l_split_chars := p_primary_separator;

```



```

l_split := instr(l_content_text, l_split_chars);
if (l_split > p_limit) or (l_split = 0 and length(l_content_text) > p_limit) then
    l_split_chars := p_secondary_separator;
    l_split := instr(l_content_text, l_split_chars);
    if (l_split > p_limit) or (l_split = 0 and length(l_content_text) > p_limit) then
        l_split_chars := '';
        l_split := p_limit;
    end if;
end if;
if l_split = 0 then
    l_chunk := trim(l_content_text);
    if length(l_chunk) > 0 then
        insert into kb_chunks(document_id, seq, chunk, split_chars) values(p_id, l_seq,
    end if;
    exit;
else
    l_chunk := trim(substr(l_content_text, 1, l_split));
    -- dbms_output.put_line('START CHUNK');
    -- dbms_output.put_line(l_chunk);
    insert into kb_chunks(document_id, seq, chunk, split_chars) values(p_id, l_seq, l_c
    l_content_text := substr(l_content_text, l_split+length(l_split_chars));
    l_seq := l_seq + 1;
end if;
end loop;
end split_into_chunks;

/**
 * Generate each embedding from chunks.
 */
procedure generate_embeddings(
    p_id in number
    ,p_collection_name in varchar2
    ,p_model_name      in varchar2
    ,p_endpoint        in varchar2
    ,p_cred_id         in varchar2
)
as
    l_count    number;
    l_request clob;
    l_request_json json_object_t;
    l_texts      json_array_t;
    l_response clob;
    l_response_json json_object_t;
    l_data        json_array_t;
    l_embedding_obj json_object_t;
    l_embedding    json_array_t;
    l_embedding_clob clob;

```

```

l_chunk_id kb_chunks.id%type;
e_llm_embed_failed exception;
begin
while true
loop
    /* exit if no candidate for generating embedding. */
    select count(*) into l_count from kb_chunks where embedding is null and document_id = p
    if l_count = 0 then
        exit;
    end if;
    /* select only 1 chunk to generate embeddings. */
    apex_collection.create_or_truncate_collection(p_collection_name);
    l_count := 1;
    for r in (
        select id, chunk from kb_chunks where embedding is null and document_id = p_id
    )
    loop
        apex_collection.add_member(
            p_collection_name => p_collection_name
            ,p_n001 => r.id
            ,p_clob001 => r.chunk
        );
        l_count := l_count + 1;
        if l_count > 1 then
            exit;
        end if;
    end loop;
    apex_collection.resequence_collection(p_collection_name);
    /* create a request body for OpenAI embedding. */
    l_texts := json_array_t();
    for r in (
        select clob001 from apex_collections where collection_name = p_collection_name orde
    )
    loop
        l_texts.append(r.clob001);
    end loop;
    l_request_json := json_object_t();
    l_request_json.put('model', p_model_name);
    l_request_json.put('input', l_texts);
    l_request := l_request_json.to_clob;
    apex_debug.info(l_request);
    /* call OpenAI Embedding */
    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 => fals
    l_response := apex_web_service.make_rest_request(
        p_url => p_endpoint || '/v1/embeddings'

```

```

        ,p_http_method => 'POST'
        ,p_body => l_request
        ,p_credential_static_id => p_cred_id
        ,p_transfer_timeout => C_OPENAI_API_TIMEOUT
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info(l_response);
        raise e_llm_embed_failed;
    end if;
    /* update embedding */
    l_response_json := json_object_t(l_response);
    l_data := l_response_json.get_array('data');
    for i in 1..l_data.get_size
    loop
        l_embedding_obj := json_object_t(l_data.get(i-1));
        l_embedding := l_embedding_obj.get_array('embedding');
        l_embedding_clob := l_embedding.to_clob;
        select n001 into l_chunk_id from apex_collections where collection_name = p_collect
        update kb_chunks set embedding = l_embedding_clob where id = l_chunk_id;
    end loop;
end loop;
end generate_embeddings;

/**
 * store embeddings in vector database.
 */
procedure upsert_vectors(
    p_id in number
    ,p_index in varchar2
    ,p_pinecone_cred in varchar2
)
as
    l_count number;
    l_request clob;
    l_request_json json_object_t;
    l_vectors json_array_t;
    l_vector json_object_t;
    l_embedding json_array_t;
    l_response clob;
    l_response_json json_object_t;
    C_UPSERT constant varchar2(100) := p_index || '/vectors/upsert';
    e_upsert_vectors_failed exception;
begin
    while true
    loop
        /* exit if all embeddings are stored. */
        select count(*) into l_count from kb_chunks

```

```

where document_id = p_id and embedding is not null and (is_indexed is null or is_indexed
if l_count = 0 then
    exit;
end if;
/* store 90 embeddings in single upsert request */
l_count := 1;
l_vectors := json_array_t();
for r in (
    select id, embedding from kb_chunks
    where embedding is not null and (is_indexed is null or is_indexed <> 'Y')
)
loop
    l_vector := json_object_t();
    l_vector.put('id', to_char(r.id));
    l_vector.put('values', json_array_t(r.embedding));
    l_vectors.append(l_vector);
    update kb_chunks set is_indexed = 'Y' where id = r.id;
    l_count := l_count + 1;
    if l_count > 90 then
        exit;
    end if;
end loop;
l_request_json := json_object_t();
l_request_json.put('vectors', l_vectors);
l_request := l_request_json.to_clob;
/*
 * call psert api of Pinecone - vectore database
 */
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 => p_pinecone_cred
);
if apex_web_service.g_status_code <> 200 then
    apex_debug.info(l_response);
    raise e_upsert_vectors_failed;
end if;
end loop;
end upsert_vectors;

/**
 * Qusetion and Answer
 */

```

```

procedure ask(
    p_question      in varchar2
    ,p_prompt_system in varchar2
    ,p_top_k        in number
    ,p_index        in varchar2
    ,p_answer       out varchar2
    ,p_question_id  out number
    ,p_score_limit  in number
    ,p_model_name   in varchar2
    ,p_endpoint     in varchar2
    ,p_cred_id      in varchar2
    ,p_pinecone_cred in varchar2
    ,p_generate_model_name in varchar2
    ,p_temperature  in number
    ,p_max_tokens   in number
    ,p_prompt_template in clob
)
as
    l_request      clob;
    l_request_json json_object_t;
    l_response     clob;
    l_response_json json_object_t;
    l_data         json_array_t;
    l_embedding_obj json_object_t;
    l_embedding     json_array_t;
    l_embedding_clob clob;
    l_question_id   kb_questions.id%type;
    /* Pinecone */
    l_matches json_array_t;
    l_vector  json_object_t;
    /* search result */
    l_chunk_id varchar2(400);
    l_score    number;
    C_QUERY constant varchar2(80) := p_index || '/query';

    l_messages json_array_t;
    l_content_system json_object_t;
    l_content_user   json_object_t;

    l_prompt      kb_responses.prompt%type;
    l_context_str clob;
    l_generations json_array_t;
    l_generated_answer kb_responses.generated_answer%type;

    l_choices json_array_t;
    l_message json_object_t;

```

```

e_query_vectors_failed exception;
e_llm_embed_failed      exception;
e_llm_generate_failed   exception;
e_bad_prompt_type       exception;
begin
  /* generate embedding from question */
  select json_object(
    key 'input'      value p_question
    ,key 'model'     value p_model_name
  returning clob) into l_request from dual;
  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 => p_endpoint || '/v1/embeddings'
    ,p_http_method => 'POST'
    ,p_body => l_request
    ,p_credential_static_id => p_cred_id
    ,p_transfer_timeout => C_OPENAI_API_TIMEOUT
  );
  if apex_web_service.g_status_code <> 200 then
    apex_debug.info(l_response);
    raise e_llm_embed_failed;
  end if;
  l_response_json := json_object_t.parse(l_response);
  l_data          := l_response_json.get_array('data');
  l_embedding_obj := json_object_t(l_data.get(0));
  l_embedding      := l_embedding_obj.get_array('embedding');
  l_embedding_clob := l_embedding.to_clob;
  /* store question in table KB_QUESTIONS. */
  insert into kb_questions(question, embedding) values(p_question, l_embedding_clob)
  returning id into l_question_id;
  p_question_id := l_question_id;
  /*
   * query pinecone by embedding generated from the question.
   */
  l_request_json := json_object_t();
  l_request_json.put('includeValues', false);
  l_request_json.put('includeMetadata', false);
  l_request_json.put('vector', l_embedding);
  l_request_json.put('topK', p_top_k);
  l_request := l_request_json.to_clob;
  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_l_static_id => p_pinecone_cred
    );
if apex_web_service.g_status_code <> 200 then
    apex_debug.info(l_response);
    raise e_query_vectors_failed;
end if;
/* store response from Pinecone in table KB_ANSWERS. */
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_chunk_id    := l_vector.get_string('id');
    l_score       := l_vector.get_number('score');
    insert into kb_answers(question_id, chunk_id, score) values(l_question_id, l_chunk_id,
end loop;
/*
 * Create Prompt for OpenAI chat completions.
 */
l_context_str := '';
for r in (
    select c.chunk from kb_answers a join kb_chunks c on a.chunk_id = c.id
    where a.question_id = l_question_id and a.score > p_score_limit
    order by a.score desc
)
loop
    l_context_str := l_context_str || r.chunk;
end loop;
l_prompt := p_prompt_template;
l_prompt := replace(l_prompt, '{context_str}', l_context_str);
l_prompt := replace(l_prompt, '{query_str}', p_question);
/*
 * call Cohere generate with the prompt.
 */
l_request_json := json_object_t();
l_messages := json_array_t();
l_content_system := json_object_t();
l_content_system.put('role','system');
l_content_system.put('content', p_prompt_system);
l_content_user := json_object_t();
l_content_user.put('role','user');
l_content_user.put('content', l_prompt);
l_messages.append(l_content_system);
l_messages.append(l_content_user);
l_request_json.put('messages', l_messages);

```

```

l_request_json.put('temperature', p_temperature);
l_request_json.put('max_tokens', p_max_tokens);
l_request_json.put('model', p_generate_model_name);
l_request := l_request_json.to_clob;
apex_debug.info(l_request);
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 => p_endpoint || '/v1/chat/completions'
    ,p_http_method => 'POST'
    ,p_body => l_request
    ,p_credential_static_id => p_cred_id
    ,p_transfer_timeout => C_OPENAI_API_TIMEOUT
);
if apex_web_service.g_status_code <> 200 then
    apex_debug.info(l_response);
    raise e_llm_generate_failed;
end if;
l_response_json := json_object_t(l_response);
l_choices := l_response_json.get_array('choices');
l_message := json_object_t(l_choices.get(0)).get_object('message');
l_generated_answer := l_message.get_string('content');
/*
 * store reponse generated by OpenAI chat/completions for further review.
 */
insert into kb_responses(question_id, iteration, prompt, generated_answer)
values(l_question_id, 1, l_prompt, l_generated_answer);
p_answer := l_generated_answer;
end ask;

/* delete vectors from Pinecone */
procedure delete_vectors(
    p_id in number
    ,p_index in varchar2
    ,p_pinecone_cred in varchar2
)
as
    l_request      clob;
    l_request_json json_object_t;
    l_vectors      json_array_t;
    l_response     clob;
    C_DELETE constant varchar2(100) := p_index || '/vectors/delete';
    e_delete_vectors_failed exception;
begin
    l_vectors := json_array_t();
    for r in (select id from kb_chunks where document_id = p_id)

```



```

loop
    l_vectors.append(to_char(r.id));
end loop;
l_request_json := json_object_t();
l_request_json.put('ids', l_vectors);
l_request_json.put('deleteAll', 'false');
l_request := l_request_json.to_clob;
/*
 * Pineconeのdelete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_DELETE
    ,p_http_method => 'POST'
    ,p_body => l_request
    ,p_credential_l_static_id => p_pinecone_cred
);
if apex_web_service.g_status_code <> 200 then
    apex_debug.info(l_response);
    raise e_delete_vectors_failed;
end if;
/* delete chunks of the document from kb_chunks. */
delete from kb_chunks where document_id = p_id;
end delete_vectors;
end kb_llm_util;
/

```

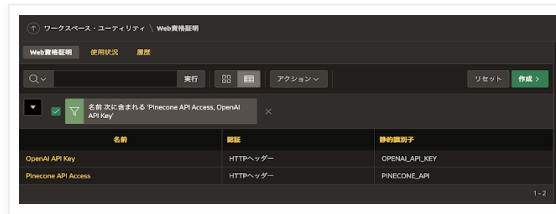
kb_llm_util_openai.sql hosted with ❤ by GitHub

[view raw](#)

アプリケーション定義の置換のセクションで、**G_INDEX**にPineconeのインデックス、**G_ENDPOINT**にOpenAI互換APIを呼び出すエンドポイント（OpenAIであれば <https://api.openai.com>）を設定します。

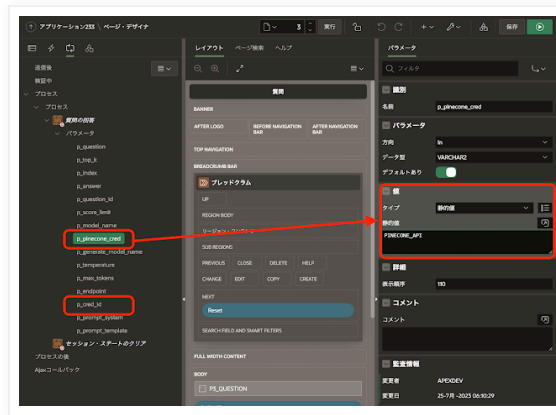


Pineconeのインデックス操作に使用する**Web資格証明**と、（OpenAIのAPIを呼び出す場合は）**OpenAIのAPI**を呼び出す際に使用する**Web資格証明**を作成しておきます。

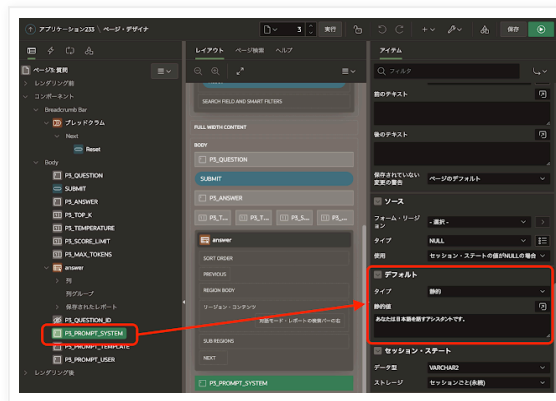


パッケージKB_LLM_UTILのプロシージャで、Pineconeにアクセスするものには引数 `p_pinecode_cred`、OpenAIのAPIアクセスを行なうものは引数 `p_cred_id` を持っています。

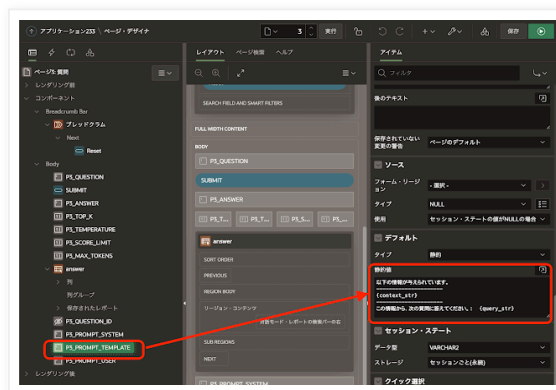
それぞれパラメータを選択し、値に作成したWeb資格証明の静的IDを設定します。



roleがsystemのメッセージのデフォルトは、ページ・アイテムP3_PROMPT_SYSTEMに設定されています。



roleがuserのメッセージを作るテンプレートは、ページ・アイテムP3_PROMPT_TEMPLATEに設定されています。



Llama2の処理に時間がかかるため、デフォルトの状態だとNginxでタイムアウトが発生します。
server.confを以下に変更し、プロキシ関連のタイムアウトを6分に延長します。

```
server {
    listen 443 ssl;
    ssl_certificate      /etc/letsencrypt/live/ホスト名/fullchain.pem;
    ssl_certificate_key  /etc/letsencrypt/live/ホスト名/privkey.pem;
    server_name ホスト名;
    root /usr/share/nginx/html;
    index index.html;

    location / {
        proxy_pass http://localhost:8000/;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_connect_timeout 360;
        proxy_send_timeout 360;
        proxy_read_timeout 360;
    }
}
```

server.conf hosted with ❤️ by GitHub

[view raw](#)

アプリケーションの変更点は以上になります。

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: 15:50

共有

<

ホーム

>

[ウェブ バージョンを表示](#)

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](#)

