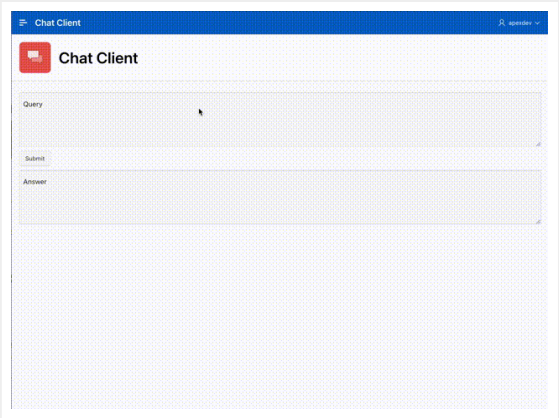
日日是Oracle APEX

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3月9日木曜日

LlamaIndexにChatGPT(gpt-3.5-turbo)を使ったチャット・サーバーを作る

以前に作成したOracle Cloudのコンピュート・インスタンスにLlamaIndexをインストールしてチャット・サーバーを作成し、Oracle APEXのアプリケーションから呼び出してみました。



以下の記事を参照しています。

Llama Index（GPT Index）にDevelopersIOの記事を100件読み込ませて質問してみた
https://dev.classmethod.jp/articles/llama-index-developersio-articles/
LlamaIndex で ChatGPT API を試す
https://note.com/npaka/n/ncbb858cf11c3
LangChain の チャットモデル (ChatGPTの新しい抽象化) を試す
https://note.com/npaka/n/n403fc29a02c7
LlamaIndex クイックスタートガイド
https://note.com/npaka/n/n8c3867a55837

チャット・サーバーの実行には、Oracle CloudのAlways Free（無料）で作成できるAmpere A1の4OCPU, 24GBのコンピュート・インスタンスを使用しています。

このサーバーは、以前にOpenAIのWisperを実装したときに作成したものです。

OpenAI Whisperを使った文字起こしアプリの作成(2) - UbuntuへのWhisperの実装
http://apexugj.blogspot.com/2023/01/openai-whisper-2-ubuntu.html
OpenAI Whisperを使った文字起こしアプリの作成(3) - Flaskを使ったAPIサーバー
http://apexugj.blogspot.com/2023/01/openai-whisper-3-api.html

Wisperの実装は不要ですが、コンピュート・インスタンスへのホスト名の割り当て（DNSでIPアドレスの解決ができる）および、サーバー証明書の発行までは準備作業として必要です。

以上の準備ができているところから作業を始めます。

Ubuntuのインスタンスにsshでログインします。作業はすべてホーム・ディレクトリ上で行います。

llama-indexをインストールします。

pip install llama-index

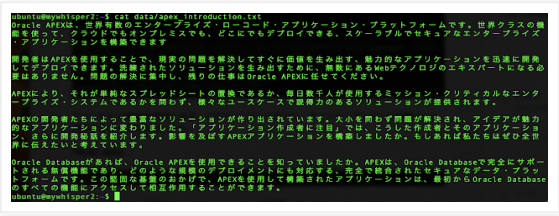
```
ubuntu@mywhisper2:~$ pip install llama-index
Processing
./.cache/pip/wheels/57/26/ea/fc7cbfb104ac458cb33bfd2610d0e5ca597b184af49e73c1bf/lla
ma_index-0.4.23-py3-none-any.whl
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages
(from llama-index) (1.23.5)
Requirement already satisfied: langchain in ./.local/lib/python3.8/site-packages
(from llama-index) (0.0.104)


[中略]

Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from
python-dateutil>=2.8.1->pandas->llama-index) (1.14.0)
Requirement already satisfied: idna>=2.0 in /usr/lib/python3/dist-packages (from
yarl<2.0,>=1.0->aiohttp<4.0.0,>=3.8.3->langchain->llama-index) (2.8)
Installing collected packages: llama-index
Successfully installed llama-index-0.4.23
ubuntu@mywhisper2:~$
```

ディレクトリ**data**を作成します。このディレクトリの下に、チャット・サーバーが知識として使用するデータを配置します。

mkdir data

```
ubuntu@mywhisper2:~$ mkdir data
ubuntu@mywhisper2:~$
```

ディレクトリ**data**の下にapex_introduction.txtを配置しました。Oracle APEXの紹介が記述されています。このデータはこの通りでなくても全く問題ないので、それぞれ興味のあるデータを配置されると良いでしょう。



ファイル**chat-server.py**として、以下の内容が記述されたファイルを作成します。index.jsonがあれば作成済みのインデックスとして読み込みます。無い場合はインデックスの作成を行います。そのため、初回実行時は起動に時間がかかります。また、ディレクトリdata以下の内容を更新した場合は、index.jsonを削除してchat-servcer.pyを再起動する必要があります。

/chatに**POST**されたJSONドキュメント**{ "query": "問合せ文字列" }**を受け取って、応答であるJSONドキュメント**{ "answer": "応答となる文字列" }**を返します。

```python
import logging
import sys
import os
import json

from llama_index import GPTSimpleVectorIndex, SimpleDirectoryReader
from llama_index.langchain_helpers.chatgpt import ChatGPTLLMPredictor
from flask import Flask, request

# ログレベルと出力先の設定
logging.basicConfig(stream=sys.stdout, level=logging.INFO, force=True)
logging.getLogger().addHandler(logging.StreamHandler(stream=sys.stdout))

# インデックスの作成。
# ファイルindex.jsonがないときだけdataディレクトリの内容を元に
# 索引を作成する。
index_file = "index.json"
index = None
if os.path.exists(index_file):
    index = GPTSimpleVectorIndex.load_from_disk(index_file)
else:
    llm_predictor = ChatGPTLLMPredictor(
        # systemメッセージを変更すると、回答の口調を変更できる。
        prepend_messages = [
            {"role": "system", "content": "You are a helpful assistant."},
            ]
        )
    #
    documents = SimpleDirectoryReader('data').load_data()
    index = GPTSimpleVectorIndex(
        documents=documents,
        llm_predictor=llm_predictor
        )
    index.save_to_disk(index_file)

# /chatの呼び出しに対する処理
app = Flask(__name__)
@app.route('/chat', methods=['POST'])
def chat():
    answer_dict = {}
    if request.method == 'POST':
        query = request.json['query']
        print("received query", query)
        answer = index.query(query)
        print("generated answer", answer)
        if answer.response:
            answer_dict["answer"] = str(answer)
```

```
    answer_json = json.dumps(answer_dict, ensure_ascii=False)
    # answer_json = json.dumps(answer_dict)
    print("response string", answer_json)
    return answer_json

if __name__ == "__main__":
        app.run(host='0.0.0.0', port=8443, ssl_context=('./certs/fullchain.pem', './certs/privk
```

chat-server.py hosted with ❤️ by GitHub view raw

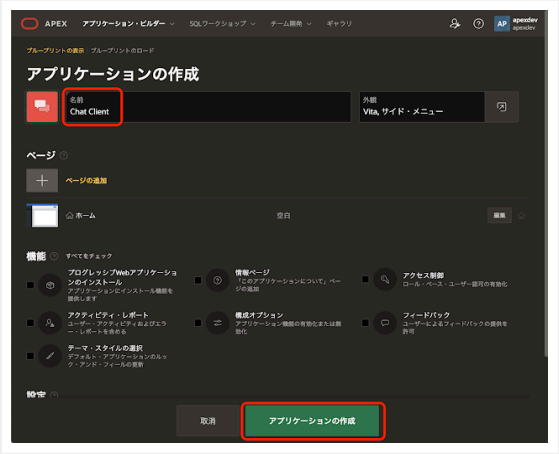
環境変数**OPENAI_API_KEY**に**OpenAIのAPIキー**を設定して、**chat-server.py**を実行します。あらかじめディレクトリ**certs**の下に**fullchain.pem**、**privkey.pem**を作成しておきます。

```
ubuntu@mywhisper2:~$ export OPENAI_API_KEY=OpenAIのAPIキー
ubuntu@mywhisper2:~$ python chat-server.py
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning:
urllib3 (1.26.13) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
 * Serving Flask app 'chat-server'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production
deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on https://127.0.0.1:8443
 * Running on https://10.0.0.131:8443
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on https://127.0.0.1:8443
 * Running on https://10.0.0.131:8443
INFO:werkzeug:Press CTRL+C to quit
Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
 * Restarting with stat
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning:
urllib3 (1.26.13) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
WARNING:werkzeug: * Debugger is active!
 * Debugger is active!
INFO:werkzeug: * Debugger PIN: 197-750-656
 * Debugger PIN: 197-750-656
```

以上でチャット・サーバーが実行されました。

このサーバーに質問するOracle APEXのアプリケーションを作成します。

アプリケーション作成ウィザードを起動し、空のアプリケーションを作成します。**名前**は**Chat Client**とします。



アプリケーションが作成されたら**アプリケーション定義**を開き、置換文字列**G_ENDPOINT**として、**チャット・サーバーの呼び出しURL**を設定します。以下のようなURLになります。

https://チャット・サーバーのホスト名/chat

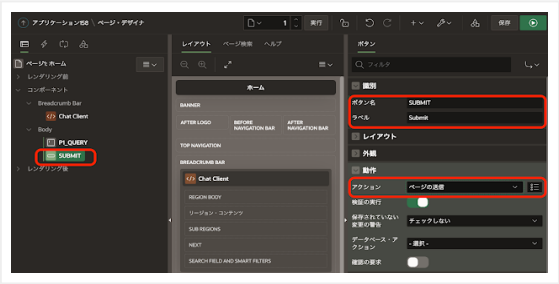
ページ・デザイナで**ホーム・ページ**を開きます。

問合せ文字列を入力するページ・アイテムを作成します。

識別の**名前**は**P1_QUERY**、**タイプ**は**テキスト領域**、**ラベル**は**Query**とします。

問合せを送信するボタンを作成します。

識別の**ボタン名**は**SUBMIT**、**ラベル**は**Submit**、**動作**の**アクション**はデフォルトの**ページの送信**とします。



チャット・サーバーが返す応答を印刷するページ・アイテムを作成します。

識別の**名前**は**P1_ANSWER**、**タイプ**は**テキスト領域**、**ラベル**は**Answer**とします。

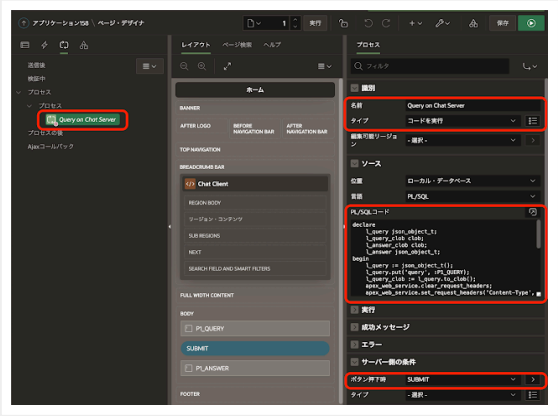
プロセス・ビューを開き、チャット・サーバーにリクエストを送信するプロセスを作成します。

識別の**名前**は**Query on Chat Server**、**タイプ**は**コードの実行**を選択します。**ソース**の**PL/SQLコード**として以下を記述します。

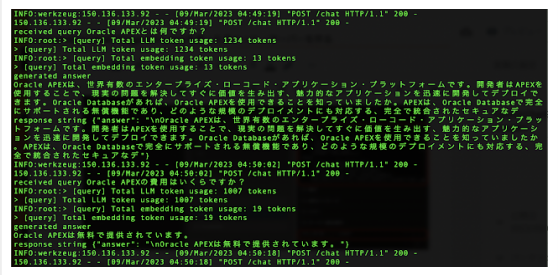
```
declare
    l_query json_object_t;
    l_query_clob clob;
    l_answer_clob clob;
    l_answer json_object_t;
begin
    l_query := json_object_t();
    l_query.put('query', :P1_QUERY);
    l_query_clob := l_query.to_clob();
    apex_web_service.clear_request_headers;
    apex_web_service.set_request_headers('Content-Type', 'application/json');
    l_answer_clob := apex_web_service.make_rest_request(
        p_url => :G_ENDPOINT
        ,p_http_method => 'POST'
        ,p_body => l_query_clob
    );
    if apex_web_service.g_status_code <> 200 then
        apex_debug.info('Chat Server Error %s, %s',
            apex_web_service.g_status_code
            ,l_answer_clob);
        raise_application_error(-20001, 'Chat Server Error = '
            || apex_web_service.g_status_code);
    end if;
    l_answer := json_object_t(l_answer_clob);
    :P1_ANSWER := l_answer.get_string('answer');
end;
```

query-on-chat-server.sql hosted with ❤️ by GitHub　view raw

サーバー側の条件の**ボタン押下時**として、**SUBMIT**を選択します。



以上でアプリケーションは完成です。実行すると記事の最初のGIF動画のように動作します。サーバー側では、以下のようなメッセージが表示されます。



クラスChatGPTLLMPredictorの引数prepend_messagesとして、ロール（system, user, assistant）の異なる履歴が設定できるようですが、それも含めてGPTSimpleVectorIndexでインデックスを作成するコードになっています。ChatGPTのように履歴を含めて問合せをする場合、毎回インデックスの作成が必要になり効率が良くないように見えます。今のところ、どうすればよいか分かりません。

今回作成したアプリケーションのエクスポートを以下に置きました。
https://github.com/ujnak/apexapps/blob/master/exports/llama-index-chat-client.zip

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: 14:44

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