OpenVINO™ 生成 AI API を使用して数行のコードで高速な生成 AI アプリを作成する方法

この記事は、Medium に公開されている「How to Build Faster GenAI Apps with Fewer Lines of Code using OpenVINO™ GenAI API」の日本語参考訳です。原文は更新される可能性があります。原文と翻訳文の内容が異なる場合は原文を優先してください。

> 新しい OpenVINO™ 生成 AI API は、開発者に、シンプルで理解しやすいコードを提供します。OpenVINO™ は、コンピューター・ビジョンと AI の高速化および最適化ライブラリーから、開発者が生成 AI を実現する手段へと進化したのです。

ChatGPT* などのチャットボットの台頭が世界を席巻する中、Generative Pre-trained Transformer (GPT) は、開発者の間でその名を知られるようになりました。生成系 AI (生成 AI) の開発、特に大規模言語モデルとチャットボットの進歩は、急速かつ絶えず変化しており、次にどのようなブレークスルーが起こるのか、開発者が何に注目すべきなのかを予測することは困難です。生成 AI はさらなる普及が見込まれていて、開発者は AI アプリケーションをローカルで開発、保守、デプロイするための理解しやすい簡単な方法を求めています。

生成 AI に対する期待にもかかわらず、特にエッジデバイスと AI PC では、これらのモデルの推論を実行することは大きな課題です。

インテルのハードウェアで最新の生成 AI を実行

現在、インテルのハードウェアで生成 AI の最高のパフォーマンスを得るには、開発者は Optimum Intel と OpenVINO™ バックエンドで最適化された Hugging Face パイプラインを使用して生成 AI のモデルを実行しています。OpenVINO™ を利用すると、CPU、GPU、NPU の最適化が可能になり、大幅にレイテンシーが削減され、効率が向上します (英語)。さらに、量子化や重み圧縮などのモデルの最適化手法を利用して、メモリー・フットプリントを最小限に抑えることができます (メモリー使用量が 1/2 から 1/3 になります)。クライアントやエッジデバイスには通常 32GB 以下のメモリーしか搭載されていないため、モデルをデプロイする際にメモリーがボトルネックになることがよくあります。

図 1: 新しい OpenVINO™ 生成 AI API を使用すると、コーディング側でさらなる改善が可能になります。推論コードは 3 行のコードに削減されます。この新しいワークフローにより、開発者が生成 AI アプリ開発の旅を始めるための学習曲線が大幅に緩和されます。

OpenVINO™ 生成 AI ライブラリーをインストールすると、コードの行数が削減されるだけでなく、インストールされる依存関係も少ないため、わずか 216MB の簡潔でコンパクトな環境で生成 AI 推論を実行できます。

図 2: OpenVINO™ 生成 AI API を使用してソリューションをデプロイすると、ディスク使用量が削減されるだけでなく、開発者が生成 AI アプリケーションの保守を行う際の大きな課題の 1 つである、生成系 AI アプリを構築するための依存関係の要件も簡素化されます。

	OpenVINO™ 生成 AI API	Optimum Intel
追加の依存関係	非常に少ない	PyTorch* などを含む 多くの依存関係
推論コード	簡単なコーディング	Hugging Face ベース
API	Python*、C++	Python*
モデルのサポート	LLM (追加の生成モデル を準備中)	広範なモデル
サンプリングの手法	グリーディー、 ビーム検索および 多項デコード	広範な手法
並列生成最適化	はい† (連続バッチ、 PagedAttention など...)	いいえ
トークン化	OpenVINO™ トークナイザーのモデル	Hugging Face トークナイザー (Python* 向け)
		†機能は開発中

表 1: OpenVINO™ 生成 AI API と Optimum Intel パッケージの比較

Optimum Intel と比較すると、生成 AI API は、グリーディーおよびビーム検索を含む、最も使用されているサンプリングの手法のみ統合します。開発者は、多項デコードを利用してサンプリング・パラメーター (Top-K や Temperature など) をカスタマイズすることもできます。

複数のユーザーのシナリオを考慮して、生成 AI API では連続バッチ (Continuous batching) と PagedAttention のみネイティブに実装しました。テキスト生成中、これらのテクノロジーは、複数のバッチで推論する際のパフォーマンスの向上とメモリー消費の最適化に役立ちます。

Hugging Face のトークナイザーは Python* でのみ動作するため、生成 AI API は、OpenVINO™ C++ ランタイムの入力/出力テンソル形式に合わせるために、2 つの OpenVINO™ モデルを個別に推論して入力テキストをトークン化し、出力ベクトルを逆トークン化します。このアプローチの前に、開発者は Optimum Intel CLI を使用して Hugging Face のトークナイザーを OpenVINO™ IR モデルに変換できます。

ここまで、新しい OpenVINO™ 生成 AI API を使用する主な利点を説明しました。次のセッションでは、デモを実行する方法をステップごとに説明します。

OpenVINO™ 生成 AI API を使用した軽量の生成 AI

インストール

生成系 AI と LLM で推論を実行するための新しい OpenVINO™ 生成 AI API のセットアップは、単純で分かりやすいように設計されています。インストール・プロセスは、PyPI またはアーカイブをダウンロードして実行できるため、ニーズに合った方法を柔軟に選択できます。例えば、最新 OpenVINO™ 2024.2 リリースに付属する PyPI インストールの場合、次のコマンドを使用します。

```
python -m pip install openvino-genai
```

インストールの詳細は、こちら (英語) を参照してください。

推論の実行

OpenVINO™ をインストールしたら、生成 AI モデルと LLM モデルで推論の実行を開始できます。この API を利用すると、数行のコードで、モデルをロードし、コンテキストを渡して、応答を受け取ることができます。内部的には、OpenVINO™ は入力テキストのトークン化を処理し、選択したデバイスで生成ループを実行して、最終的な応答を提供します。openvino.genai リポジトリーで提供されている chat_sample (英語) に基づいて、Python* と C++ の両方でプロセスをステップごとに確認してみましょう。

ステップ 1: Hugging Face Optimum Intel (この例では、チャット用に調整された Tiny Llama) を使用して、LLM モデルをダウンロードし、OpenVINO™ IR 形式にエクスポートします。このステップでは、依存関係の競合を回避するため、別の仮想環境を作成することを推奨します。以下に例を示します。

```
python -m venv openvino_venv
```

アクティベートします。

```
openvino_venv\Script\activate
```

モデルのエクスポート・プロセスに必要な依存関係をインストールします。依存関係は openvino.genai リポジトリーのここから (英語) 入手できます。

```
python -m pip install --upgrade-strategy eager -r requirements.txt
```

モデルをダウンロードしてエクスポートするには、次のコマンドを使用します。

```
optimum-cli export openvino --trust-remote-code --model TinyLlama/TinyLlama-
1.1B-Chat-v1.0 TinyLlama-1.1B-Chat-v1.0
```

LLM 推論時のパフォーマンスを向上するには、モデルの重みに INT4 などの低い精度を使用することを推奨します。以下に示すように、モデルのエクスポート・プロセス中にニューラル・ネットワーク圧縮フレームワーク (NNCF) を使用して重みを圧縮できます。

```
optimum-cli export openvino --trust-remote-code --model TinyLlama/TinyLlama-
1.1B-Chat-v1.0 TinyLlama-1.1B-Chat-v1.0 --weight-format int4
```

モデルをエクスポートする必要があるのは 1 回のみで、このステップでインストールした仮想環境と依存関係はもう必要ないため、この仮想環境をディスクから削除してもかまいません。

ステップ 2: Python* または C++ API で LLM のテキスト生成の推論を実行します。

新しい Python* API でパイプラインを設定:

```
pipe = ov_genai.LLMPipeline(model_path, "CPU")
print(pipe.generate("The Sun is yellow because"))
```

新しい C++ API でパイプラインを設定:

```
int main(int argc, char* argv[]) {
std::string model_path = argv[1];
ov::genai::LLMPipeline pipe(model_path, "CPU");//target device is CPU
std::cout << pipe.generate("The Sun is yellow because"); //input context
```

LLM 生成パイプラインの構築に必要なのは、数行のコードのみです。これだけ単純なのは、Hugging Face Optimum Intel からエクスポートされたモデルに、**トークナイザー/逆トークナイザー**や生成の構成など、実行に必要なすべての情報がすでに含まれているためであり、Hugging Face の生成と一貫性のある結果が保証されます。依存関係とアプリケーションへの追加を最小限に抑えて、LLM を実行する C++ と Python* の両方の API を提供しています。

提供されるコードは CPU 上で動作しますが、デバイス名を「GPU」に置き換えることで GPU 上で動作させることもできます。

```
pipe = ov_genai.LLMPipeline(model_path, "GPU")
```

生成向けのさらにインタラクティブな UI を作成するため、モデル出力トークンのストリーミングのサポートを追加しました。ストリーマーから **True** を返すことにより、いつでもトークン生成を停止できます。

テキスト生成の推論にはステートフル・モデル (英語) を内部で実行して、生成速度を高速化し、データ表現の変換によるオーバーヘッドを軽減しています。そのため、入力間で KVCache を維持すると有益です。次の例のように、チャット固有の start_chat と finish_chat メソッドを使用して会話セッションをマークします。

Python* の例:

```
import argparse
import openvino_genai


def streamer(subword):
    print(subword, end='', flush=True)
    # Return flag corresponds to whether generation should be stopped.
    # False means continue generation.
    return False
model_path = TinyLlama-1.1B-Chat-v1.0


device = 'CPU'  # GPU can be used as well
pipe = openvino_genai.LLMPipeline(args.model_dir, device)
```

```
config = openvino_genai.GenerationConfig()
config.max_new_tokens = 100


pipe.start_chat()
while True:
    prompt = input('question:\n')
    if 'Stop!' == prompt:
        break
    pipe.generate(prompt, config, streamer)


    print('\n----------')
pipe.finish_chat()
```

C++ の例:

```
#include "openvino/genai/llm_pipeline.hpp"


int main(int argc, char* argv[]) try {
    if (2 != argc) {
        throw std::runtime_error(std::string{"Usage: "} + argv[0]
                                 + " <MODEL_DIR>");
    }
    std::string prompt;
    std::string model_path = argv[1];


    std::string device = "CPU";  // GPU can be used as well
    ov::genai::LLMPipeline pipe(model_path, "CPU");

    ov::genai::GenerationConfig config;
    config.max_new_tokens = 100;
    std::function<bool(std::string)> streamer = [](std::string word) {
        std::cout << word << std::flush;
        // Return flag corresponds to whether generation should be stopped.
        // false means continue generation.
        return false;
    };


    pipe.start_chat();
    for (;;) {
        std::cout << "question:\n";

        std::getline(std::cin, prompt);
        if (prompt == "Stop!")
            break;


        pipe.generate(prompt, config, streamer);

        std::cout << "\n----------\n";
    }
    pipe.finish_chat();
```

```
} catch (const std::exception& error) {
    std::cerr << error.what() << '\n';
    return EXIT_FAILURE;
} catch (...) {
    std::cerr << "Non-exception object thrown\n";
    return EXIT_FAILURE;
}
```

最後に、上記の例を AI PC で実行した結果を示します。

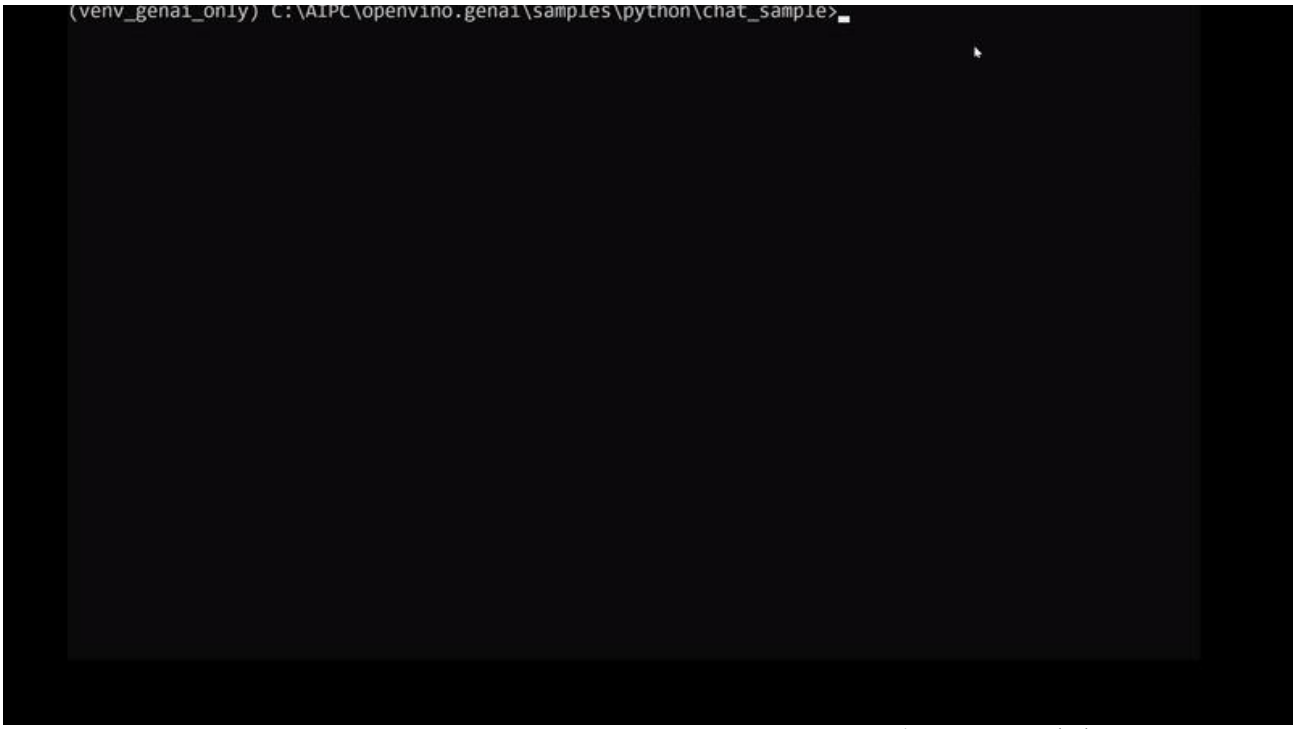

図 3: AI PC でローカルに実行した Llama ベースのチャットボットのライブデモ。

生成 AI API には、軽量のデプロイとコーディングを可能にする次の API が含まれています。

- generation_config — 生成されるテキストの最大長、文末トークンを無視するかどうか、デコード手法の詳細 (グリーディー、ビーム検索、多項サンプリング) など、生成プロセスをカスタマイズするための設定です。
- llm_pipeline — 入力の処理、テキストの生成、設定可能なオプションによる出力の管理のパイプラインを含む、テキスト生成向けのクラスとユーティリティーを提供します。
- streamer_base — ストリーマーを作成するための抽象基本クラスです。
- tokenizer — テキストのエンコードとデコードのためのトークナイザー・クラスです。
- visibility — 生成 AI ライブラリーの可視性を制御します。

まとめ

最新の OpenVINO™ 2024.2 リリースの新しい OpenVINO™ 生成 AI API は、多くの利点と機能を提供する、開発者が生成 AI と LLM アプリケーションを作成するための強力なツールです。セットアップ・プロセスが簡単で依存関係が少ないこの API を利用すると、コードの複雑さが軽減され、数行のコードで効率の良い生成 AI 推論パイプラインを迅速に構築できます。また、ストリーミング・モデル出力トークンのサポートにより、インタラクティブな UI の作成が容易になり、ユーザー・エクスペリエンスが向上します。

新しい生成 AI API を試し、プロジェクトでその機能を利用してみてください。オープンソースのライブラリーを利用して、生成系 AI が到達できる新しい境地を開きましょう。

参考文献 (英語)

https://huggingface.co/docs/transformers/pipeline_tutorial

関連情報 (英語)

OpenVINO™ ドキュメント
OpenVINO™ ノートブック
フィードバックの提供 & 問題の報告

貢献者および編集者 (英語)

Ria Cheruvu、Paula Ramos、Ryan Loney、Stephanie Maluso

OpenVINO™ ツールキットとは

AI を加速する無償のツールである OpenVINO™ ツールキットは、インテルが無償で提供しているインテル製の CPU や GPU、VPU、FPGA などのパフォーマンスを最大限に活用して、コンピューター・ビジョン、画像関係をはじめ、自然言語処理や音声処理など、幅広いディープラーニング・モデルで推論を最適化し高速化する推論エンジン/ツールスイートです。

OpenVINO™ ツールキット・ページでは、ツールの概要、利用方法、導入事例、トレーニング、ツール・ダウンロードまでさまざまな情報を提供しています。ぜひ特設サイトにアクセスしてみてください。

https://www.intel.co.jp/content/www/jp/ja/internet-of-things/openvino-toolkit.html