C# アプリに LLM を組み込もう

竹内一希

自己紹介

自己紹介

- 名前
  - 竹内 一希
- 所属
  - 近畿大学工学部 電子情報工学科
  - 近畿大学 マイコン部
- 好きな技術
  - C#
  - dotnet
  - コンパイラ技術
- SNS
  - X:@_actbit
  - GitHub:actbit

はじめに

はじめに

皆さん LLM って使ってますか？

LLM って Python しか使えないって思っていませんか???

実は C# からでも動かせるんです

なのでその方法についてお話していきます！

使えるライブラリ

使えるライブラリ

以下のライブラリが使用できました

- LLamaSharp
- Aspire でコンテナ管理

LLamaSharp

LLamaSharp は **llama.cpp** を C# 向けに wrap したものです。なので llama2,llama3 ベースや Qwen1.5 ベースのライブラリが使用できます！

実際に動かしていきます！

開発環境の構築

1. dotnet をインストールします！

dotnet のインストール

Windows

```
winget install Microsoft.DotNet.SDK.9
```

Linux(Ubuntu 系)

```
sudo apt update
sudo apt install dotnet-sdk-8.0
```

LLamaSharp (ii)

プロジェクトを作成

プロジェクト

```
dotnet new console LLamaSharpSample
```

プロジェクトに移動

```
cd LLamaSharpSample.csproj
```

LLamaSharp (iii)

ライブラリを導入

LLamaSharp を導入

```
dotnet add package LLamaSharp --version 0.23.0
```

環境に応じて変更

CPU

```
dotnet add package LLamaSharp.Backend.Cpu --version 0.23.0
```

Cuda12

```
dotnet add package LLamaSharp.Backend.Cuda12 --version 0.23.0
```

Vulkan

```
dotnet add package LLamaSharp.Backend.Vulkan --version 0.23.0
```

LLamaSharp (iv)

実際のコード

必要な namespace の読み込み

```
using LLama;
using LLama.Common;
using LLama.Sampling;
using LLama.Transformers;
```

コードで guff ファイルの読み込み等を行います。

```
var modelPath = Console.ReadLine().Trim('\"');
var parameters = new ModelParams(modelPath)
{
    GpuLayerCount = 15
};
using var model = LLamaWeights.LoadFromFile(parameters);
using var context = model.CreateContext(parameters);
var ex = new InteractiveExecutor(context);
```

LLamaSharp (v)

chathistory の作成

```
ChatHistory chatHistory = new ChatHistory();
chatHistory.AddMessage(AuthorRole.System, "あなたは優秀なアシスタントです。どんなことでも的確に答える必要があり、間違えることは許されません。しかしながら、間違えてしまった場合には即座に認め訂正してください");
chatHistory.AddMessage(AuthorRole.User, "こんにちは");
chatHistory.AddMessage(AuthorRole.Assistant, "お手伝いが必要ですか？");
```

LLamaSharp (vi)

コード生成時の設定を作成

```
var inferenceParams = new InferenceParams
{
    SamplingPipeline = new DefaultSamplingPipeline
    {
        Temperature = 0.9f
    },
    AntiPrompts = new List<string> { "User:" },
    MaxTokens=-1
};
```

LLamaSharp (vii)

ChatSession の作成及び設定

```
var chatSession = new ChatSession(ex, chatHistory);
chatSession.WithHistoryTransform(new PromptTemplateTransformer(model,
withAssistant: true));
chatSession.WithOutputTransform(new
LLamaTransforms.KeywordTextOutputStreamTransform(
    ["User:", "🦙"],
    redundancyLength: 5));
```

LLamaSharp (viii)

実際にコードを生成させる

```
while (true)
{
    Console.Write("User>");
    string prompt = Console.ReadLine();
    Console.Write("Assistant>");

    await foreach (var text in chatSession.ChatAsync(
        new ChatHistory.Message(AuthorRole.User, prompt), inferenceParams))
    {
        Console.ForegroundColor = ConsoleColor.White;
        Console.Write(text);
    }
}
```

LLamaSharp を使って作成中の作品

AITDD 開発ツール

- C# にはドキュメントコメントという機能がある。ドキュメントコメントの機能によりコメント上にドキュメントと同様の情報を書くことができる。
- コードにはメソッドなどのパブリックメンバ定義 + 多量のドキュメントコメント
- 実際にしてほしい挙動のテストコードを作成する
- LLM がコードの未実装部を実装
- Build と Test を行い問題があれば LLM にフィードバックし、再度コード生成を行う
  - 正しい挙動になるまで繰り返す

LLamaSharp を使って作成中の作品 (ii)

実際には

以下のようなコードから

```
/// <summary>
///  計算するためのクラス
/// </summary>
class calc
{
  /// <summary>
  /// <paramref name="n1"/> + <paramref name="n2"/>の計算を行います。
  /// </summary>
  /// <param name="n1">一つ目の数値</param>
  /// <param name="n2">二つ目の数値</param>
  /// <returns><paramref name="n1"/>+<paramref name="n2"/></returns>
  public int Sum(int n1,int n2)
  {
      throw new NotImplementedException();
  }
}
```

LLamaSharp を使って作成中の作品 (iii)

以下のようなコードを生成する

```
/// <summary>
///  計算するためのクラス
/// </summary>
class calc
{
  /// <summary>
  /// <paramref name="n1"/> + <paramref name="n2"/>の計算を行います。
  /// </summary>
  /// <param name="n1">一つ目の数値</param>
  /// <param name="n2">二つ目の数値</param>
  /// <returns><paramref name="n1"/>+<paramref name="n2"/></returns>
  public int Sum(int n1,int n2)
  {
      return n1 + n2;
  }
}
```

一部のドキュメントをとても重視する SI のような開発では重宝できるのではないか？