Large Language Model Application Development on Microsoft Windows* 11 using Intel-BigDL Python*

Development Guide

September 2023

Document Number: 787707-0.5

Contents

Tables

Revision History

Date	Revision	Description
September 2023	0.5	Initial release.

§

1.0 Introduction

As the demand for flexibility in AI-generated content (AIGC) increases, recent years have witnessed a rise in the demand for large language model (LLM) applications to generate creative content. Users may prefer LLM applications on local devices like PCs for personal data.

This document provides the solution for large language model application development on Windows 11 PC using Gradio and Intel BigDL. The CPU platform prefers 13th Core™ i7 and i9 (or future) with 16 GB system memory, which is user-friendly and supports 16 billion parameters (16B) INT4 large language model.

BigDL seamlessly scales your data analytics and AI applications from laptop to cloud, with LLM: Low-bit (INT3/INT4/INT5/INT8) large language model library for Intel CPU/GPU and so on.

Gradio is the fast way to demo your machine learning model with a friendly web interface so that anyone can use it anywhere.

§

2.0 Setup Environment and Development

This section explains an environment setup and LLM application development on Microsoft Windows* 11. Application support models like:

- ChatGLM2-6B: Chinese-English chat large language model
- LLaMa2-13B: English chat large language model
- StarCoder-15.5B: Chinese-English code generation large language model

2.1 Install Intel-BigDL on Microsoft Windows* 11

1. Install Miniconda3-py39_23.5.2-0-Windows-x86_64.exe on Windows 11, download package from https://docs.conda.io/en/latest/miniconda.html#windows-installers
2. Open Anaconda Powershell Prompt and type the following commands.

```
conda create -n llm python=3.9

conda activate llm

pip install --pre --upgrade bigdl-llm[all]

pip install gradio mdtex2html
```

or install the BigDL special version. Must be bigdl-llm version 2.4.0b20230820 or latest.

```
pip install --pre --upgrade bigdl-llm[all]==2.4.0b20230820 -i
https://pypi.tuna.tsinghua.edu.cn/simple
```

2.2 Model Conversion and Load Model by Python*

For example, download ChatGLM2, LLaMa2, and StarCoder FP16 models from hugging face.

- ChatGLM2-6B：https://huggingface.co/THUDM/chatglm2-6b/tree/main
- LLaMa2-13B: https://huggingface.co/meta-llama/Llama-2-13b-chat-hf/tree/main
- StarCoder-15.5B: https://huggingface.co/bigcode/starcoder/tree/main

Open Anaconda PowerShell Prompt, modify the model path, and execute the following command to convert the FP16 model to native INT4.

```
conda activate llm

llm-convert "C:/llm-models/chatglm2-6b/" --model-format pth --model-
family "chatglm" --outfile "checkpoint/"

llm-convert "C:/llm-models/llama-2-13b-chat-hf/" --model-format pth --
model-family "llama" --outfile "checkpoint/"

llm-convert  "C:/llm-models/starcoder/" --model-format pth --model-family
"starcoder" --outfile "checkpoint/"
```

Note: StarCoder needs minimum 16GB system memory to convert FP16 to the native INT4 model.

Load native INT4 model by Python.

- The parameter of n_threads depends on Core platform. Please try the following choices and pick the best one.
    - n_threads=CPU P-cores *2 + E-cores **OR**
    - n_threads=CPU P-cores *2 + E-cores - 1 **OR**
    - n_threads=CPU P-cores *2 + E-cores - 2
- n_ctx=4096 means the longest tokens of the sum of input and output is 4096 tokens.

```
from bigdl.llm.ggml.model.chatglm.chatglm import ChatGLM
from bigdl.llm.transformers import BigdlNativeForCausalLM
model_name = "chatglm2-6b"
model_all_local_path = "C:\\PC_LLM\\checkpoint\\"
if model_name == "chatglm2-6b":
    model = ChatGLM(model_all_loc*al_path + "\\ggml-chatglm2-6b-
q4_0.bin", n_threads=20,n_ctx=4096)

elif model_name == "llama2-13b":
    model = BigdlNativeForCausalLM.from_pretrained(
    pretrained_model_name_or_path=model_all_local_path +
"\\bigdl_llm_llama2_13b_q4_0.bin",
    model_family='llama',n_threads=20,n_ctx=4096)
elif model_name == "StarCoder":
    model = BigdlNativeForCausalLM.from_pretrained(
    pretrained_model_name_or_path=model_all_local_path +
"\\bigdl_llm_starcoder_q4_0.bin",
    model_family='starcoder',n_threads=20,n_ctx=4096)
```

2.3 Large Language Model Benchmark on CPU

Test native INT4 LLM benchmark on CPU to get the best performance data.

1. Open Anaconda PowerShell Prompt

```
conda activate llm
```

2. ChatGLM2:

```
llm-cli -t 20 -x chatglm -m "ggml-chatglm2-6b-q4_0.bin" -p "Once upon a
time, there existed a little girl who liked to have adventures. She
wanted to go to places and meet new people, and have fun" --no-mmap -v -n
32
```

3. LLaMa2:

```
llm-cli -t 20 -x llama -m "bigdl_llm_llama2_13b_q4_0.bin" -p "Once upon a
time, there existed a little girl who liked to have adventures. She
wanted to go to places and meet new people, and have fun" --no-mmap -n 32
```

4. StarCoder:

```
llm-cli -t 20 -x starcoder -m "bigdl_llm_starcoder_q4_0.bin" -p "Once
upon a time, there existed a little girl who liked to have adventures.
She wanted to go to places and meet new people, and have fun" --no-mmap -
n 32
```

Parameters:

- Input content "Once upon a time, there existed a little girl who liked to have adventures. She wanted to go to places and meet new people and have fun" has 32 tokens.
- -n 32 means output 32 tokens.

Get performance data from the command line as shown in Figure 1.

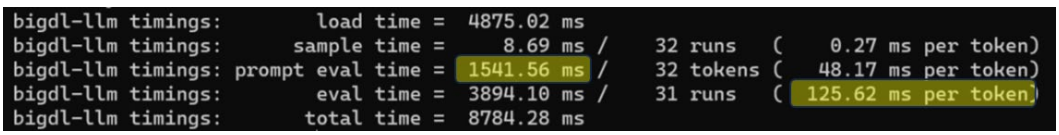
Input token: 32 tokens

Output token: 32 tokens (31 runs = 32 tokens – 1st token)

1st token avg latency (ms) = 1541.56 ms

2nd+ token avg latency (ms/token) = 125.62 ms per token

Figure 1. LLM Performance Data



2.4 Large Language Model Stream Chat in Application

ChatGLM2, LLaMa2, and StarCoder are the same “for” functions to stream chat.

Parameters:

- Temperature: The higher the value, the more random the output. The adjustable range is 0~1.
- Top P: The higher its value, the greater the diversity of word choices, adjustable range 0~1.
- max_tokens: The maximum tokens for the output text, the adjustable range is 1~2048. The model determines the upper limit. For example, the maximum n_ctx of these three models is 8k, so the sum of input and output tokens should be less than 8k.

```
from bigdl.llm.ggml.model.chatglm.chatglm import ChatGLM
model_name = "chatglm2-6b"
model_all_local_path = "C:\\PC_LLM\\checkpoint\\"
prompt = "What is AI?"
if model_name == "chatglm2-6b":
    model = ChatGLM(model_all_local_path + "\\ggml-chatglm2-6b-q4_0.bin",
n_threads=20,n_ctx=4096)
    response = ""
    for chunk in model(prompt,
temperature=0.95,top_p=0.8,stream=True,max_tokens=512):
        response += chunk['choices'][0]['text']
```

2.5 Add Multi-Chat History Feature in Application

Add Multi-Chat History Feature based on code of chapter 2.4.

```
from bigdl.llm.ggml.model.chatglm.chatglm import ChatGLM
model_name = "chatglm2-6b"
model_all_local_path = "C:\\PC_LLM\\checkpoint\\"
history_round = 0
history = []
if model_name == "chatglm2-6b":
    model = ChatGLM(model_all_local_path + "\\ggml-chatglm2-6b-q4_0.bin",
n_threads=20,n_ctx=4096)
input = "你好"
predict(input)
input = "请进行丽江三天必游景点旅游规划"
predict(input)

def predict(input):
```

```
    global history_round, model, history
    response = ""
    if len(model.tokenize(history)) > 2500 or history_round >= 5: ###
history record 5 rounds
        history_round = 0
        history = []
        print("********** reset chatbot and history", history)

    if len(history) == 0:
        print("********** new chat ")
        prompt = input
        history = prompt
        history_round = 1
    else:
        prompt = history + '\n' + input
        history_round += 1
    print("****************** history_round ", history_round)

    timeStart = time.time()
    for chunk in model(prompt, temperature=0.9,top_p=0.9,
stream=True,max_tokens=512):
        response += chunk['choices'][0]['text']
    history = prompt + response
    print("******** max_length history",len(model.tokenize(history)))
```

2.6 Develop Application by Gradio Web UI

1. Users need an administrator to open Anaconda PowerShell Prompt to use all CPU cores.
2. Run the LLM_demo_v1.0.py or LLM_demo_v2.0.py script to open the LLM Application UI, as shown in Figure 2.

```
git clone https://github.com/KiwiHana/LLM_UI_Windows_CPU.git

conda activate llm

python LLM_demo_v1.0.py
```

Note: modify LLM_demo_v1.0.py Line 289 model path in the main function.

For example, model_all_local_path = "C:/Users/username/checkpoint/"

Figure 2. LLM Application UI

- Large language model application has the following files:
  - LLM_demo_v1.0.py
  - theme3.json
  - checkpoint
    - bigdl_llm_llama2_13b_q4_0.bin
    - bigdl_llm_starcoder_q4_0.bin
    - ggml-chatglm2-6b-q4_0.bin

§

3.0 Conclusion

This document consists explanation of large language model application development on Microsoft Windows 11 PC using Gradio and Intel BigDL, where large language model includes ChatGLM2-6B, LLaMa2-13B and StarCoder-15.5B models.

§

4.0 Reference Documents

Table 1. Reference Documents

Document	Document Location
BigDL	https://github.com/intel-analytics/BigDL
BigDL Tutorial	https://github.com/intel-analytics/bigdl-llm-tutorial/tree/main
LLM Application on Windows	https://github.com/KiwiHana/LLM_UI_Windows_CPU

§