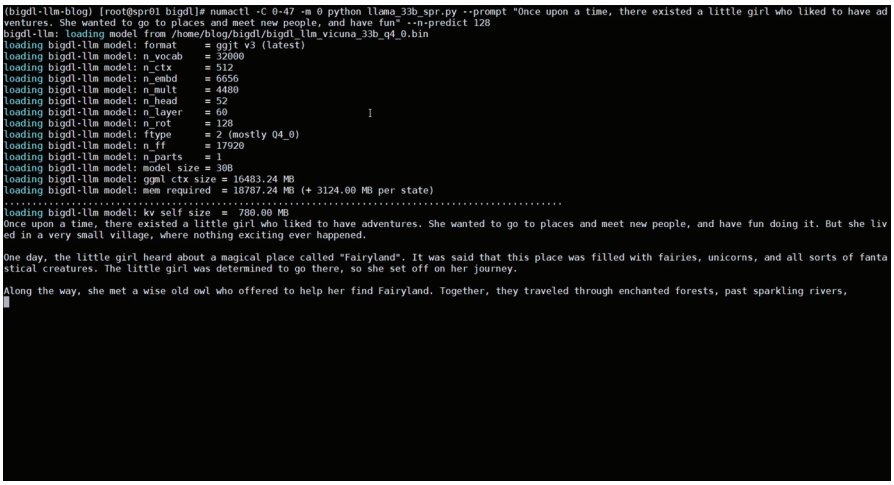
BigDL-LLM：在英特尔® 平台上加速大语言模型的便捷新利器

作者：英特尔公司黄晟盛、黄凯、戴金权

我们正迈入一个由大语言模型 (Large Language Model, LLM) 驱动的 AI 新时代，LLM 在诸如客户服务、虚拟助理、内容创作、编程辅助等各类应用中正发挥着越来越重要的作用。然而，随着 LLM 规模不断扩大，运行大模型所需的资源消耗也越来越大，导致其运行也越来越慢，这给 AI 应用开发者带来了相当大的挑战。为此，英特尔最近推出了一个名为 BigDL-LLM[i] 的大模型开源库，可助力 AI 开发者和研究者在英特尔® 平台上加速优化大语言模型，提升大语言模型在英特尔® 平台上的使用体验。

下面就展示了使用 BigDL-LLM 加速过的 330 亿参数的大语言模型 Vicuna-33b-v1.3[ii] 在一台搭载英特尔® 至强® 铂金 8468 处理器的服务器上运行的实时效果。



视频：在一台搭载英特尔® 至强® 铂金 8468 处理器的服务器上运行 330 亿参数大语言模型的实际速度（实时录屏）

BigDL-LLM：英特尔® 平台上的开源大语言模型加速库

BigDL-LLM 是一个针对大语言模型的优化加速库，是开源 BigDL 的一部分，通过 Apache 2.0 许可证发布。它提供了各种低精度优化（例如 INT4/INT5/INT8），并可利用多种英特尔® CPU 集成的硬件加速技术（AVX/VNNI/AMX 等）和最新的软件优化，来赋能大语言模型在英特尔® 平台上实现更高效的优化和更为快速的运行。

BigDL-LLM 的一大重要特性是：对基于 Hugging Face Transformers API 的模型，只需改动一行代码即可对模型进行加速，理论上可以支持运行**任何** Transformers 模型，这对熟悉 Transformers API 的开发者非常友好。除了 Transformers API，很多人也会使用 LangChain 来开发大语言模型应用。为此，BigDL-LLM 也提供便于使用的 LangChain 的集成[iii]，从而让开发者能够轻松使用 BigDL-

LLM 来开发新应用或迁移现有的、基于 Transformers API 或 LangChain API 的应用。此外，对于一般的 PyTorch 大语言模型（没有使用 Transformer 或 LangChain API 的模型），也可使用 BigDL-LLM `optimize_model` API 一键加速来提升性能。详情请参阅 GitHub README[iv] 以及官方文档[v]。

BigDL-LLM 还提供了大量常用开源 LLM 的加速样例（e.g. 使用 Transformers API 的 样例[vi] 和使用 LangChain API 的样例[vii]，以及教程（包括配套 jupyter notebooks）[viii]，方便开发者快速上手尝试。

安装和使用：简便的安装过程和易用的 API 接口

安装 BigDL-LLM 非常简便，只需执行如下所示的这一行命令即可。

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

使用 BigDL-LLM 对大模型进行加速也是非常容易的（这里仅用 Transformers 风格 API 进行举例）。使用 BigDL-LLM Transformer 风格 API 对模型加速，只需要改动模型加载部分，后续使用过程与原生 Transformers 完全一致。而用 BigDL-LLM API 加载模型的方式与 Transformers API 也几乎一致——用户只需要更改 import，在 from_pretrained 参数中设置 **load_in_4bit=True** 即可。BigDL-LLM 会在加载模型的过程中对模型进行 4-bit 低精度量化，并在后续推理过程中利用各种软硬件加速技术优化其执行。

```
#Load Hugging Face Transformers model with INT4 optimizations
from bigdl.llm.transformers import AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained('/path/to/model/', load_in_4bit=True)
```

示例：快速实现一个基于大语言模型的语音助手应用

下文将以 LLM 常见应用场景"语音助手"为例，展示采用 BigDL-LLM 快速实现 LLM 应用的案例。通常情况下，语音助手应用的工作流程分为以下两个部分：

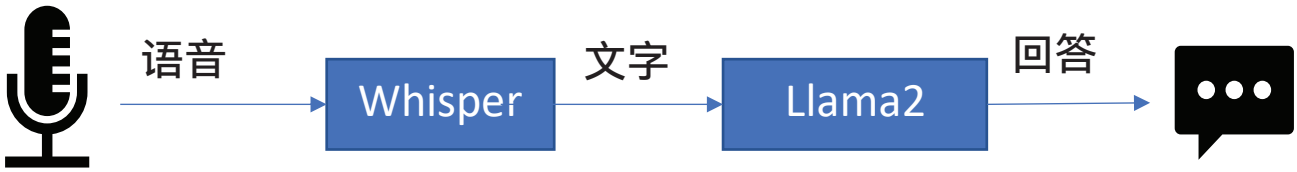


图 1. 语音助手工作流程示意

1、语音识别——使用语音识别模型（本示例采用了 Whisper 模型[ix]）将用户的语音转换为文本；
2、文本生成——将 1 中输出的文本作为提示语（prompt），使用一个大语言模型（本示例采用了 Llama2[x]）生成回复。

以下是本文使用 BigDL-LLM 和 LangChain[xi] 来搭建语音助手应用的过程：

在**语音识别**阶段：第一步，加载预处理器 processor 和语音识别模型 recog_model。本示例中使用的识别模型 Whisper 是一个 Transformers 模型。只需使用 BigDL-LLM 中的 **AutoModelForSpeechSeq2Seq** 并设置参数 **load_in_4bit=True**，就能够以 INT4 精度加载并加速这一模型，从而显著缩短模型推理用时。

```
processor = WhisperProcessor .from_pretrained(recog_model_path)
recog_model = AutoModelForSpeechSeq2Seq .from_pretrained(recog_model_path, load_in_4bit=True)
```

第二步，进行语音识别。首先使用处理器从输入语音中提取输入特征，然后使用识别模型预测 token，并再次使用处理器将 token 解码为自然语言文本。

```
input_features = processor(frame_data,
                           sampling_rate=audio.sample_rate,
                           return_tensor="pt").input_features
predicted_ids = recogn_model.generate(input_features, forced_decoder_ids=forced_decoder_ids)
text = processor.batch_decode(predicted_ids, skip_special_tokens=True)[0]
```

在**文本生成**阶段，首先使用 BigDL-LLM 的 **TransformersLLM API** 创建一个 LangChain 语言模型（TransformersLLM 是在 BigDL-LLM 中定义的语言链 LLM 集成）。您可以使用这个 API 加载任何一个 Hugging Face Transformers 模型。

```
llm = TransformersLLM . from_model_id(
        model_id=llm_model_path,
        model_kwargs={"temperature": 0,
                      "max_length": args.max_length,
                      "trust_remote_code": True},
)
```

然后，创建一个正常的对话链 **LLMChain**，并将已经创建的 llm 设置为输入参数。

```
# The following code is complete the same as the use-case
voiceassistant_chain = LLMChain(
    llm=llm,
    prompt=prompt,
    verbose=True,
    memory=ConversationBufferWindowMemory(k=2),
)
```

这个链条将会记录所有的对话历史，并将对话历史适当地格式化为大语言模型的提示语，用于生成回复。这时候只需要将识别模型生成的文本作为 "human_input" 输入即可，代码如下：

```
response_text = voiceassistant_chain .predict(human_input=text,
                                              stop="\n\n")
```

最后，将语音识别和文本生成步骤放入循环中，即可在多轮对话中与该"语音助手"交谈。您可访问此链接[xii]，查看完整的示例代码，并使用自己的电脑进行尝试。快用 BigDL-LLM 来快速搭建自己的语音助手吧！

作者简介：

英特尔公司 AI 资深架构师黄晟盛，英特尔公司 AI 框架工程师黄凯，英特尔院士、大数据技术全球 CTO、BigDL 项目创始人戴

权，都在从事大数据和 AI 相关工作。

---

[i] https://github.com/intel-analytics/BigDL/tree/main/python/llm

[ii] Vicuna 模型是社区基于 LLaMA 模型微调而得的。https://huggingface.co/lmsys/vicuna-33b-v1.3

[iii] https://github.com/intel-analytics/BigDL/blob/main/python/llm/README.md#langchain-api

[iv] https://github.com/intel-analytics/BigDL/blob/main/python/llm/README.md

[v] https://bigdl.readthedocs.io/en/latest/doc/LLM/index.html

[vi] https://github.com/intel-analytics/BigDL/tree/main/python/llm/example/transformers/transformers_int4

[vii] https://github.com/intel-analytics/BigDL/tree/main/python/llm/example/langchain

[viii] https://github.com/intel-analytics/bigdl-llm-tutorial

[ix] https://github.com/openai/whisper

[x] https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

[xi] https://github.com/langchain-ai/langchain

[xii] https://github.com/intel-analytics/BigDL/blob/main/python/llm/example/langchain/transformers_int4/voiceassistant.py