llama.cpp/examples/main（这是llama.cpp中的一个例子）

This example program allows you to use various LLaMA language models easily and efficiently. It is specifically designed to work with the llama.cpp project, which provides a plain C/C++ implementation with optional 4-bit quantization support for faster, lower memory inference, and is optimized for desktop CPUs. This program can be used to perform various inference tasks with LLaMA models, including generating text based on user-provided prompts and chat-like interactions with reverse prompts.

- 这个例子程序允许你轻松的、有效的使用多种不同的llama语言模型。
- 它专为与 llama.cpp 项目配合使用而设计，该项目提供纯 C/C++ 实现，并具有可选的 4 位量化支持，可实现更快、更低内存的推理，并且针对台式机 CPU 进行了优化。
- 这个程序可以通过使用llama models被用于执行多种不同的推理任务，包含基于用户提供的提示词的生成式文本、反向提示词的对话式交互等场景应用。

Table of Contents（内容目录）

Quick Start（快速开始）

To get started right away, run the following command, making sure to use the correct path for the model you have（为了直接快速的开始，运行下列命令，对于你拥有的模型确保使用对其正确的文件路径）:

First, we will need to download a model. In these examples, we will use the Gemma model from the ggml-org repo on Hugging Face.（首先，我们我需要下载一个model。在这些例子中，我们将会使用gemma模型，这个gemma模型来自hugging face上ggml-org仓库中的一个模型） https://huggingface.co/ggml-org/gemma-1.1-7b-it-Q4_K_M-GGUF/resolve/main/gemma-1.1-7b-it.Q4_K_M.gguf?download=true

Once downloaded, place your model in the models folder in llama.cpp.（一旦下载完成之后，将下载后的模型放在llama.cpp项目中的model文件夹中）

Unix-based systems (Linux, macOS, etc.)（基于unix的系统，例如linux,macos,等等）:

Input prompt (One-and-done)（输入提示词：只输入一次即完成）

```
# 使用llama-cli程序，参数-m指定具体模型的位置，参数--prompt指定输入的提示词
# 参数1：./llama-cli是程序名称
# 参数2：-m models/gemma-1.1-7b-it.Q4_K_M.gguf是指定的模型文件
```

```
# 参数3：--prompt "Once upon a time"是提示词
./llama-cli -m models/gemma-1.1-7b-it.Q4_K_M.gguf --prompt "Once upon a time"
```

Conversation mode (Allow for continuous interaction with the model)（对话模式：允许和model进行连续的交互）

```
# 使用llama-cli程序，参数-m指定具体模型的位置，参数--cnv指定对话模式， 参数--chat-
template指定对话的模板
# 参数1：./llama-cli是程序名称
# 参数2：-m models/gemma-1.1-7b-it.Q4_K_M.gguf是指定的模型文件
# 参数3：-cnv是开启conversation mode的开关
# 参数4：--chat-template gemma这个参数不能知道具体是怎么使用的
./llama-cli -m models/gemma-1.1-7b-it.Q4_K_M.gguf -cnv --chat-template gemma
```

Infinite text from a starting prompt (you can use `Ctrl-C` to stop it)（从一个初始的提示符开始，系统或程序会持续不断地生成文本）：

```
# 使用llama-cli程序，参数-m指定具体模型的位置，剩余的两个参数还需要理解
# 参数1：./llama-cli是程序名称
# 参数2：-m models/gemma-1.1-7b-it.Q4_K_M.gguf是指定的模型文件
# 参数3：--ignore-eos是开启conversation mode的开关
# 参数4：--chat-template gemma指定同模型对话的方式
# 参数5：--ignore-eos指定-->忽略流结束标记并继续生成
# 参数6：-n -1指定预测的tokens数量，其中-1为无限的选项
./llama-cli -m models/gemma-1.1-7b-it.Q4_K_M.gguf --ignore-eos -n -1
```

Windows:

Input prompt (One-and-done)

```
# 参数1：./llama-cli.exe程序路径
# 参数2：-m models\gemma-1.1-7b-it.Q4_K_M.gguf用来指定模型文件的位置
# 参数3：--prompt "Once upon a time"用来指定模型输入的提示词
./llama-cli.exe -m models\gemma-1.1-7b-it.Q4_K_M.gguf --prompt "Once upon a time"
```

Conversation mode (Allow for continuous interaction with the model)

```
# 参数1：./llama-cli.exe程序路径
# 参数2：-m models\gemma-1.1-7b-it.Q4_K_M.gguf用来指定模型文件的位置
# 参数3：-cnv用来开启模型的“持续对话模式”
# 参数4：--chat-template gemma用来指定对话聊天的模板
./llama-cli.exe -m models\gemma-1.1-7b-it.Q4_K_M.gguf -cnv --chat-template gemma
```

Infinite text from a starting prompt (you can use `Ctrl-C` to stop it):

```
# 参数1：./llama-cli.exe程序路径
# 参数2：-m models\gemma-1.1-7b-it.Q4_K_M.gguf用来指定模型文件的位置
# 参数3：--ignore-eos指定-->忽略流结束标记并继续生成
# 参数4：-n -1指定预测的tokens数量，其中-1为无限的选项
llama-cli.exe -m models\gemma-1.1-7b-it.Q4_K_M.gguf --ignore-eos -n -1
```

Common Options（常用的选项）

In this section, we cover the most commonly used options for running the `llama-cli` program with the LLaMA models（在这部分，针对运行llama-cli程序，我们将会涵盖最为常用的选项）:

- `-m FNAME, --model FNAME`: Specify the path to the LLaMA model file （指定llama模型的路径）(e.g., `models/gemma-1.1-7b-it.Q4_K_M.gguf`; inferred from `--model-url` if set).
- `-mu MODEL_URL --model-url MODEL_URL`: Specify a remote http url to download the file （指定一个remote http url下载model文件）(e.g https://huggingface.co/ggml-org/gemma-1.1-7b-it-Q4_K_M-GGUF/resolve/main/gemma-1.1-7b-it.Q4_K_M.gguf?download=true).
- `-i, --interactive`: Run the program in interactive mode, allowing you to provide input directly and receive real-time responses.（以交互式的方式运行程序）
- `-n N, --n-predict N`: Set the number of tokens to predict when generating text. Adjusting this value can influence the length of the generated text.（当生成文本的时候设置输出tokens的数量。通过调整这个值可以影响生成文本的长短。）
- `-c N, --ctx-size N`: Set the size of the prompt context. The default is 512, but LLaMA models were built with a context of 2048, which will provide better results for longer input/inference.（设置提示词的上下文大小。默认的是512个token，但是llama模型是以2048的上下文构建的，这2048的上下文将会对更长的输入和推理提供更好的结果。）
- `-mli, --multiline-input`: Allows you to write or paste multiple lines without ending each in '' （这个参数允许你不需要要\即可输入或者拷贝多行输入文本）
- `-t N, --threads N`: Set the number of threads to use during generation. For optimal performance, it is recommended to set this value to the number of physical CPU cores your system has.（在运行的时候通过设置这个值来指定需要使用多少的threads。对于最优的性能，建议设置成物理CPU cores的数量）
- - `-ngl N, --n-gpu-layers N`: When compiled with GPU support, this option allows offloading some layers to the GPU for computation. Generally results in increased performance.（当使用GPU支持编译的时候，这个选项将会允许把一部分负载迁移到GPU上进行计算。通常情况下将会增加性能）

Input Prompts（输入提示词）

The `llama-cli` program provides several ways to interact with the LLaMA models using input prompts（这个llama-cli程序提供了使用提示词同llama models进行交互的方式）:

- `--prompt PROMPT`: Provide a prompt directly as a command-line option.（直接提供一个提示词作为命令行选项）
- `--file FNAME`: Provide a file containing a prompt or multiple prompts.（提供一个包含一个提示词或者多个提示词的文件）

- `--interactive-first`: Run the program in interactive mode and wait for input right away. (More on this below.) (以交互模式和等待输入提示词的方式运行程序)

Interaction (交互)

The `llama-cli` program offers a seamless way to interact with LLaMA models, allowing users to engage in real-time conversations or provide instructions for specific tasks. The interactive mode can be triggered using various options, including `--interactive` and `--interactive-first`.（llama-cli程序提供了一种同llama模型无缝交互的方式，这种方式允许用户以一种实时对话的方式进行交互，或者提供了对于特定任务的指令。这个交互模式可以通过使用--interactive和--interactive-first的参数被激活。)

In interactive mode, users can participate in text generation by injecting their input during the process. Users can press `Ctrl+C` at any time to interject and type their input, followed by pressing `Return` to submit it to the LLaMA model. To submit additional lines without finalizing input, users can end the current line with a backslash (`\`) and continue typing. (在交互模式下，用户可以在生成的过程中通过注入他们的输入来参与其中。用户可以在任何使用按下Ctrl_C中断生成过程并且输入他们的提示词，输入提示词之后可以通过按下Return来向llama models提交他们的提示词。在没有完成输入想要提交额外的输入，用户可以在输入行末尾添加\来延续输入。)

Interaction Options (交互选项)

- `-i, --interactive`: Run the program in interactive mode, allowing users to engage in real-time conversations or provide specific instructions to the model. (以交互模式运行程序，这种模式允许用户以实时的方式同模型进行交互或者给模型提供特定的指令)
- `--interactive-first`: Run the program in interactive mode and immediately wait for user input before starting the text generation.
- `-cnv, --conversation`: Run the program in conversation mode (does not print special tokens and suffix/prefix, use default chat template) (default: false)
- `--color`: Enable colorized output to differentiate visually distinguishing between prompts, user input, and generated text. (启动彩色的输出为了能够从视觉上区分提示词、用户输入和生成的文本)

By understanding and utilizing these interaction options, you can create engaging and dynamic experiences with the LLaMA models, tailoring the text generation process to your specific needs. (通过理解和使用这些交互选项，你可以创建一个同模型交互的、动态的体验，对于你特定的需要可以个性化文本的生成过程。)

Reverse Prompts

反向提示是一种在文本生成过程中，当遇到特定的文本字符串时暂停生成，并切换到交互模式的功能。这种机制特别适用于创建聊天式的交互体验。

Reverse prompts are a powerful way to create a chat-like experience with a LLaMA model by pausing the text generation when specific text strings are encountered:

- `-r PROMPT, --reverse-prompt PROMPT`: Specify one or multiple reverse prompts to pause text generation and switch to interactive mode. For example, `-r "User:"` can be used to jump back into the conversation whenever it's the user's turn to speak. This helps create a more interactive and conversational experience. However, the reverse prompt doesn't work when it ends with a space.

- **`-r PROMPT, --reverse-prompt PROMPT`**：这是命令行参数，用于指定一个或多个反向提示。当文本生成过程中遇到这些指定的字符串时，生成会暂停，并根据需要切换到交互模式。

    - **示例**：使用 `-r "User:"` 表示当生成的文本中出现“User:”时，模型会自动暂停输出，等待用户输入，从而实现交互。这有助于在模拟对话中创造一种流畅的交替说话体验。

- **空格问题**：如果反向提示字符串的末尾有空格，这种提示机制将不会工作。这可能是因为文本处理过程中空格被忽略或者不被识别为有效的暂停点。

To overcome this limitation, you can use the `--in-prefix` flag to add a space or any other characters after the reverse prompt.

- **`--in-prefix` 标志**：为了克服上述空格的限制，可以使用 `--in-prefix` 参数。这个标志允许在反向提示后添加空格或其他字符。这样做可以确保即使原本的反向提示以空格结束，也能正常触发暂停和交互切换。

In-Prefix

The `--in-prefix` flag is used to add a prefix to your input, primarily, this is used to insert a space after the reverse prompt. Here's an example of how to use the `--in-prefix` flag in conjunction with the `--reverse-prompt` flag:

```
./llama-cli -r "User:" --in-prefix " "
```

这种技术在实现基于命令行的交互式聊天机器人或其他需要细致控制文本输出时刻的应用中非常有用。通过合理设置反向提示，开发者可以精确地控制何时暂停文本生成，以及如何根据用户的输入动态地继续对话，从而提高用户交互的自然性和流畅性。

In-Suffix

The `--in-suffix` flag is used to add a suffix after your input. This is useful for adding an "Assistant:" prompt after the user's input. It's added after the new-line character (`\n`) that's automatically added to the end of the user's input. Here's an example of how to use the `--in-suffix` flag in conjunction with the `--reverse-prompt` flag（这个--in-suffix标志被用于在你的输入后面添加一个后缀。对于添加一个assistant是非常有用的。在用户输入完成后自动添加到输入后面。这是一个例子展示如何添加一个--in-suffix标签。）:

```
./llama-cli -r "User:" --in-prefix " " --in-suffix "Assistant:"
```

When --in-prefix or --in-suffix options are enabled the chat template (--chat-template) is disabled

Chat templates（聊天模板）

`--chat-template JINJA_TEMPLATE`: This option sets a custom jinja chat template（这个选项设置了一个自定义的聊天模板）. It accepts a string, not a file name（这个选项接收的是一个字符串而不是一个文件的名字。）. Default: template taken from model's metadata（默认情况：模板信息将会从model的元数据中获得）. Llama.cpp only supports some pre-defined templates（llama.cpp项目仅支持约定好的模板）. These include llama2, llama3, gemma, monarch, chatml, orion, vicuna, vicuna-orca, deepseek, command-r, zephyr. When --in-prefix or --in-suffix options are enabled the chat template (--chat-template) is disabled（这些模板包括llama2\llama3\gemma\monarch\charml\orion\vicuna等等）.

Example usage: `--chat-template gemma`

Context Management（上下文管理）

During text generation, LLaMA models have a limited context size, which means they can only consider a certain number of tokens from the input and generated text（在文本生成的过程中，llama模型有一个限制上下文大小，这个上下文大小意味着这些模型从输入个生成的文本中考虑的token大小。）. When the context fills up, the model resets internally, potentially losing some information from the beginning of the conversation or instructions（当上下文最大之后，这个模型内部将会自动重置，潜在的从对话或者指令中丢失一些信息。）. Context management options help maintain continuity and coherence in these situations（上下文管理选项将会在这些场景中维持连续性和连贯性。）.

Context Size（上下文大小）

- `-c N, --ctx-size N`: Set the size of the prompt context (default: 0, 0 = loaded from model). The LLaMA models were built with a context of 2048-8192, which will yield the best results on longer input/inference（参数：-c N或者--ctx-size N：这是设置提示词上下文大小的参数。llama模型的上下文大小是2024-8192，在这个范围中，模型将会产生最好的结果。）.

Extended Context Size（扩展上下文大小）

Some fine-tuned models have extended the context length by scaling RoPE. For example, if the original pre-trained model has a context length (max sequence length) of 4096 (4k) and the fine-tuned model has 32k. That is a scaling factor of 8, and should work by setting the above `--ctx-size` to 32768 (32k) and `--rope-scale` to 8.（一些微调的模型已经扩展了上下文的大小。举例来说，如果原始的预处理模型有一个4K的上下文大小，微调的模型将会有32K的上下文大小，这里的伸缩因子是8的倍数，将上下文大小设置为32K也是可以运行的）

- `--rope-scale N`: Where N is the linear scaling factor used by the fine-tuned model.（这里的N是被微调模型使用的线性因子。）

Keep Prompt（持续提示词）

The `--keep` option allows users to retain the original prompt when the model runs out of context, ensuring a connection to the initial instruction or conversation topic is maintained.（当模型用完上下文大小的时候，这个--keep选项允许用户保持原始的提示词，从而确保了与原始指令或者对话主题之间的连接性。）

- `--keep N`: Specify the number of tokens from the initial prompt to retain when the model resets its internal context. By default, this value is set to 0 (meaning no tokens are kept). Use `-1` to retain all tokens from the initial prompt.（当模型重置模型内部的上下文大小的时候，--keep N参数指定从原始提示词中获得多少的token，。默认情况下，这个值会被设置为0，这意味着没有tokens将会被选取。如果设置为-1那么意味着将会从原始的提示词中获取全部的tokens.）

By utilizing context management options like `--ctx-size` and `--keep`, you can maintain a more coherent and consistent interaction with the LLaMA models, ensuring that the generated text remains relevant to the original prompt or conversation.(通过使用上下文管理选项，例如--ctx-size和--keep，你可以同模型之间保持一个更加连贯和更加一致的交互，从而确保生成的文本和原始的提示词保持相关性。)

Generation Flags（生成标签）

The following options allow you to control the text generation process and fine-tune the diversity, creativity, and quality of the generated text according to your needs. By adjusting these options and experimenting with different combinations of values, you can find the best settings for your specific use case.（下列的选项允许你控制文本生成过程和多样性的、创造性的、高质量的微调。通过调整这些选项和进行不同值得组合你可以发现你需要得最好得设置。）

Number of Tokens to Predict（预测token的数量）

- `-n N, --predict N`: Set the number of tokens to predict when generating text (default: -1, -1 = infinity, -2 = until context filled)（设置模型预测的tokens数量）

The `--predict` option controls the number of tokens the model generates in response to the input prompt. By adjusting this value, you can influence the length of the generated text. A higher value will result in longer text, while a lower value will produce shorter text.（--predict选项控制模型对输入提示词能够预测的token数量。通过调整这个值，你可以影响生成文本的长度。有个更高的值将会产生更长的文本，而一个更低的值将会生成一个更短的生成文本。）

A value of -1 will enable infinite text generation, even though we have a finite context window. When the context window is full, some of the earlier tokens (half of the tokens after `--keep`) will be discarded. The context must then be re-evaluated before generation can resume. On large models and/or large context windows, this will result in a significant pause in output.（-1将会无限的生成文本，即使我们有一个有限的上下文窗口。当上下文窗口满了，一个更早的tokens将会被丢弃。在生成的结果被使用的时候，上下文必须被重新评估。在一个更大的模型和更大的上下文窗口中，这个将会导致输出被有效的暂停。）

If the pause is undesirable, a value of -2 will stop generation immediately when the context is filled.

It is important to note that the generated text may be shorter than the specified number of tokens if an End-of-Sequence (EOS) token or a reverse prompt is encountered. In interactive mode, text generation will pause and control will be returned to the user. In non-interactive mode, the program will end. In both cases, the text generation may stop before reaching the specified `--predict` value. If you want the model to keep going without ever producing End-of-Sequence on its own, you can use the `--ignore-eos` parameter.（重要的是要注意，如果遇到序列结束（EOS）标记或反向提示，生成的文本可能会比指定的令牌数更短。在交互模式中，文本生成将暂停并将控制权返回给用户。在非交互模式中，程序将结束。在这两种情况下，文本生成可能在达到指定的 `--predict` 值之前停止。如果您希望模型继续运行而不自动产生序列结束标记，您可以使用 `--ignore-eos` 参数。）

Temperature

- `--temp N`: Adjust the randomness of the generated text (default: 0.8).

Temperature is a hyperparameter that controls the randomness of the generated text. It affects the probability distribution of the model's output tokens. A higher temperature (e.g., 1.5) makes the output more random and creative, while a lower temperature (e.g., 0.5) makes the output more focused, deterministic, and conservative. The default value is 0.8, which provides a balance between randomness and determinism. At the extreme, a temperature of 0 will always pick the most likely next token, leading to identical outputs in each run.

Example usage: `--temp 0`

Repeat Penalty

- `--repeat-penalty N`: Control the repetition of token sequences in the generated text default: 1.0, 1.0 = disabled).
- `--repeat-last-n N`: Last n tokens to consider for penalizing repetition (default: 64, 0 = disabled, -1 = ctx-size).
- `--no-penalize-nl`: Disable penalization for newline tokens when applying the repeat penalty.

The `repeat-penalty` option helps prevent the model from generating repetitive or monotonous text. A higher value (e.g., 1.5) will penalize repetitions more strongly, while a lower value (e.g., 0.9) will be more lenient. The default value is 1.

The `repeat-last-n` option controls the number of tokens in the history to consider for penalizing repetition. A larger value will look further back in the generated text to prevent repetitions, while a smaller value will only consider recent tokens. A value of 0 disables the penalty, and a value of -1 sets the number of tokens considered equal to the context size (`ctx-size`).

Use the `--no-penalize-nl` option to disable newline penalization when applying the repeat penalty. This option is particularly useful for generating chat conversations, dialogues, code, poetry, or any text where newline tokens play a significant role in structure and formatting. Disabling newline penalization helps maintain the natural flow and intended formatting in these specific use cases.

Example usage: `--repeat-penalty 1.15 --repeat-last-n 128 --no-penalize-nl`

Top-K Sampling

- `--top-k N`: Limit the next token selection to the K most probable tokens (default: 40).

Top-k sampling is a text generation method that selects the next token only from the top k most likely tokens predicted by the model. It helps reduce the risk of generating low-probability or nonsensical tokens, but it may also limit the diversity of the output. A higher value for top-k (e.g., 100) will consider more tokens and lead to more diverse text, while a lower value (e.g., 10) will focus on the most probable tokens and generate more conservative text. The default value is 40.

Example usage: `--top-k 30`

Top-P Sampling

- `--top-p N`: Limit the next token selection to a subset of tokens with a cumulative probability above a threshold P (default: 0.9).

Top-p sampling, also known as nucleus sampling, is another text generation method that selects the next token from a subset of tokens that together have a cumulative probability of at least p. This method provides a balance between diversity and quality by considering both the probabilities of tokens and the number of tokens to sample from. A higher value for top-p (e.g., 0.95) will lead to more diverse text, while a lower value (e.g., 0.5) will generate more focused and conservative text. The default value is 0.9.

Example usage: `--top-p 0.95`

Min-P Sampling

- `--min-p N`: Sets a minimum base probability threshold for token selection (default: 0.1).

The Min-P sampling method was designed as an alternative to Top-P, and aims to ensure a balance of quality and variety. The parameter *p* represents the minimum probability for a token to be considered, relative to the probability of the most likely token. For example, with *p*=0.05 and the most likely token having a probability of 0.9, logits with a value less than 0.045 are filtered out.

Example usage: `--min-p 0.05`

Tail-Free Sampling (TFS)

- `--tfs N`: Enable tail free sampling with parameter z (default: 1.0, 1.0 = disabled).

Tail-free sampling (TFS) is a text generation technique that aims to reduce the impact of less likely tokens, which may be less relevant, less coherent, or nonsensical, on the output. Similar to Top-P it tries to determine the bulk of the most likely tokens dynamically. But TFS filters out logits based on the second derivative of their probabilities. Adding tokens is stopped after the sum of the second derivatives reaches the parameter z. In short: TFS looks at how quickly the probabilities of the tokens decrease and cuts off the tail of unlikely tokens using the parameter z. Typical values for z are in the range of 0.9 to 0.95. A value of 1.0 would include all tokens and thus disables the effect of TFS.

Example usage: `--tfs 0.95`

Locally Typical Sampling

- `--typical N`: Enable locally typical sampling with parameter p (default: 1.0, 1.0 = disabled).

Locally typical sampling promotes the generation of contextually coherent and diverse text by sampling tokens that are typical or expected based on the surrounding context. By setting the parameter p between 0 and 1, you can control the balance between producing text that is locally coherent and diverse. A value closer to 1 will promote more contextually coherent tokens, while a value closer to 0 will promote more diverse tokens. A value equal to 1 disables locally typical sampling.

Example usage: `--typical 0.9`

Mirostat Sampling

- `--mirostat N`: Enable Mirostat sampling, controlling perplexity during text generation (default: 0, 0 = disabled, 1 = Mirostat, 2 = Mirostat 2.0).
- `--mirostat-lr N`: Set the Mirostat learning rate, parameter eta (default: 0.1).
- `--mirostat-ent N`: Set the Mirostat target entropy, parameter tau (default: 5.0).

Mirostat is an algorithm that actively maintains the quality of generated text within a desired range during text generation. It aims to strike a balance between coherence and diversity, avoiding low-quality output caused by excessive repetition (boredom traps) or incoherence (confusion traps).

The `--mirostat-lr` option sets the Mirostat learning rate (eta). The learning rate influences how quickly the algorithm responds to feedback from the generated text. A lower learning rate will result in slower adjustments, while a higher learning rate will make the algorithm more responsive. The default value is `0.1`.

The `--mirostat-ent` option sets the Mirostat target entropy (tau), which represents the desired perplexity value for the generated text. Adjusting the target entropy allows you to control the balance between

coherence and diversity in the generated text. A lower value will result in more focused and coherent text, while a higher value will lead to more diverse and potentially less coherent text. The default value is `5.0`.

Example usage: `--mirostat 2 --mirostat-lr 0.05 --mirostat-ent 3.0`

Logit Bias

- `-l TOKEN_ID(+/-)BIAS, --logit-bias TOKEN_ID(+/-)BIAS`: Modify the likelihood of a token appearing in the generated text completion.

The logit bias option allows you to manually adjust the likelihood of specific tokens appearing in the generated text. By providing a token ID and a positive or negative bias value, you can increase or decrease the probability of that token being generated.

For example, use `--logit-bias 15043+1` to increase the likelihood of the token 'Hello', or `--logit-bias 15043-1` to decrease its likelihood. Using a value of negative infinity, `--logit-bias 15043-inf` ensures that the token `Hello` is never produced.

A more practical use case might be to prevent the generation of `\code{begin}` and `\code{end}` by setting the `\` token (29905) to negative infinity with `-l 29905-inf`. (This is due to the prevalence of LaTeX codes that show up in LLaMA model inference.)

Example usage: `--logit-bias 29905-inf`

RNG Seed

- `-s SEED, --seed SEED`: Set the random number generator (RNG) seed (default: -1, -1 = random seed).

The RNG seed is used to initialize the random number generator that influences the text generation process. By setting a specific seed value, you can obtain consistent and reproducible results across multiple runs with the same input and settings. This can be helpful for testing, debugging, or comparing the effects of different options on the generated text to see when they diverge. If the seed is set to a value less than 0, a random seed will be used, which will result in different outputs on each run.

Performance Tuning and Memory Options

These options help improve the performance and memory usage of the LLaMA models. By adjusting these settings, you can fine-tune the model's behavior to better suit your system's capabilities and achieve optimal performance for your specific use case.

Number of Threads

- `-t N, --threads N`: Set the number of threads to use during generation. For optimal performance, it is recommended to set this value to the number of physical CPU cores your system has (as opposed to the logical number of cores). Using the correct number of threads can greatly improve performance.
- `-tb N, --threads-batch N`: Set the number of threads to use during batch and prompt processing. In some systems, it is beneficial to use a higher number of threads during batch processing than during generation. If not specified, the number of threads used for batch processing will be the same as the number of threads used for generation.

Mlock

- `--mlock`: Lock the model in memory, preventing it from being swapped out when memory-mapped. This can improve performance but trades away some of the advantages of memory-mapping by requiring more RAM to run and potentially slowing down load times as the model loads into RAM.

No Memory Mapping

- `--no-mmap`: Do not memory-map the model. By default, models are mapped into memory, which allows the system to load only the necessary parts of the model as needed. However, if the model is larger than your total amount of RAM or if your system is low on available memory, using mmap might increase the risk of pageouts, negatively impacting performance. Disabling mmap results in slower load times but may reduce pageouts if you're not using `--mlock`. Note that if the model is larger than the total amount of RAM, turning off mmap would prevent the model from loading at all.

NUMA support

- `--numa distribute`: Pin an equal proportion of the threads to the cores on each NUMA node. This will spread the load amongst all cores on the system, utilitizing all memory channels at the expense of potentially requiring memory to travel over the slow links between nodes.
- `--numa isolate`: Pin all threads to the NUMA node that the program starts on. This limits the number of cores and amount of memory that can be used, but guarantees all memory access remains local to the NUMA node.
- `--numa numactl`: Pin threads to the CPUMAP that is passed to the program by starting it with the numactl utility. This is the most flexible mode, and allow arbitrary core usage patterns, for example a map that uses all the cores on one NUMA nodes, and just enough cores on a second node to saturate the inter-node memory bus.

These flags attempt optimizations that help on some systems with non-uniform memory access. This currently consists of one of the above strategies, and disabling prefetch and readahead for mmap. The latter causes mapped pages to be faulted in on first access instead of all at once, and in combination with pinning threads to NUMA nodes, more of the pages end up on the NUMA node where they are used. Note that if the model is already in the system page cache, for example because of a previous run without this option, this will have little effect unless you drop the page cache first. This can be done by rebooting the system or on Linux by writing '3' to '/proc/sys/vm/drop_caches' as root.

Memory Float 32

- `--memory-f32`: Use 32-bit floats instead of 16-bit floats for memory key+value. This doubles the context memory requirement and cached prompt file size but does not appear to increase generation quality in a measurable way. Not recommended.

Batch Size

- `-b N, --batch-size N`: Set the batch size for prompt processing (default: `2048`). This large batch size benefits users who have BLAS installed and enabled it during the build. If you don't have BLAS enabled ("BLAS=0"), you can use a smaller number, such as 8, to see the prompt progress as it's evaluated in some situations.

- `-ub N`, `--ubatch-size N`: physical maximum batch size. This is for pipeline parallelization. Default: `512`.

Prompt Caching

- `--prompt-cache FNAME`: Specify a file to cache the model state after the initial prompt. This can significantly speed up the startup time when you're using longer prompts. The file is created during the first run and is reused and updated in subsequent runs. **Note**: Restoring a cached prompt does not imply restoring the exact state of the session at the point it was saved. So even when specifying a specific seed, you are not guaranteed to get the same sequence of tokens as the original generation.

Grammars & JSON schemas

- `--grammar GRAMMAR`, `--grammar-file FILE`: Specify a grammar (defined inline or in a file) to constrain model output to a specific format. For example, you could force the model to output JSON or to speak only in emojis. See the GBNF guide for details on the syntax.

- `--json-schema SCHEMA`: Specify a JSON schema to constrain model output to (e.g. `{}` for any JSON object, or `{"items": {"type": "string", "minLength": 10, "maxLength": 100}, "minItems": 10}` for a JSON array of strings with size constraints). If a schema uses external `$ref`s, you should use `--grammar "$( python examples/json_schema_to_grammar.py myschema.json )"` instead.

Quantization

For information about 4-bit quantization, which can significantly improve performance and reduce memory usage, please refer to llama.cpp's primary README.

Additional Options

These options provide extra functionality and customization when running the LLaMA models:

- `-h, --help`: Display a help message showing all available options and their default values. This is particularly useful for checking the latest options and default values, as they can change frequently, and the information in this document may become outdated.
- `--verbose-prompt`: Print the prompt before generating text.
- `-mg i, --main-gpu i`: When using multiple GPUs this option controls which GPU is used for small tensors for which the overhead of splitting the computation across all GPUs is not worthwhile. The GPU in question will use slightly more VRAM to store a scratch buffer for temporary results. By default GPU 0 is used.
- `-ts SPLIT, --tensor-split SPLIT`: When using multiple GPUs this option controls how large tensors should be split across all GPUs. `SPLIT` is a comma-separated list of non-negative values that assigns the proportion of data that each GPU should get in order. For example, "3,2" will assign 60% of the data to GPU 0 and 40% to GPU 1. By default the data is split in proportion to VRAM but this may not be optimal for performance.
- `--lora FNAME`: Apply a LoRA (Low-Rank Adaptation) adapter to the model (implies --no-mmap). This allows you to adapt the pretrained model to specific tasks or domains.
- `--lora-base FNAME`: Optional model to use as a base for the layers modified by the LoRA adapter. This flag is used in conjunction with the `--lora` flag, and specifies the base model for the adaptation.

- `-hfr URL --hf-repo URL`: The url to the Hugging Face model repository. Used in conjunction with `--hf-file` or `-hff`. The model is downloaded and stored in the file provided by `-m` or `--model`. If `-m` is not provided, the model is auto-stored in the path specified by the `LLAMA_CACHE` environment variable or in an OS-specific local cache.