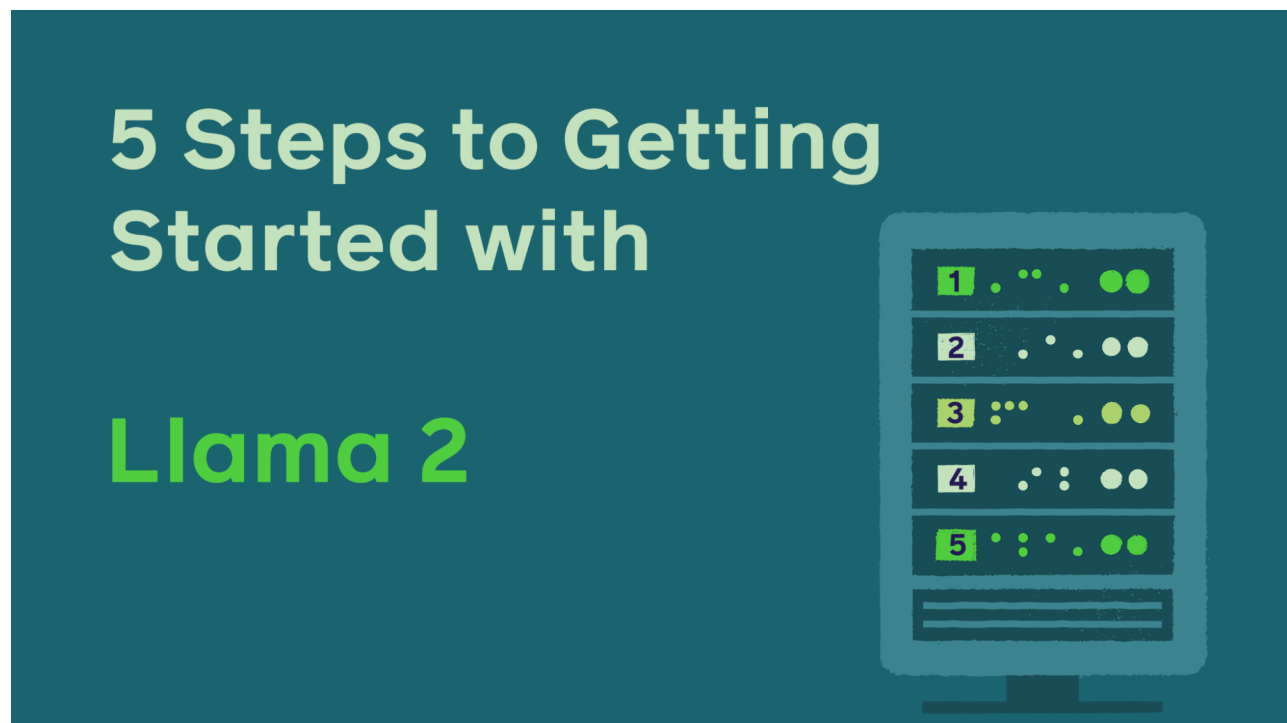


FEATURED

Large Language Model

5 Steps to Getting Started with Llama 2


November 15, 2023



The recent strides made in artificial intelligence (AI) have not only captured the curiosity of the public but have also underscored what pioneers in this field have always understood: the immense potential these technologies hold to empower us in achieving the extraordinary. This potential extends far beyond mere algorithms; it promises to usher in a new era of economic growth, social advancement, and novel modes of expression and connection.

At Meta, we strongly believe in an open approach to AI development, particularly within the dynamic landscape of generative AI. By sharing AI models openly, we extend their benefits to all corners of society. We recently [open sourced Llama 2](https://ai.meta.com/blog/5-steps-to-getting-started-with-llama-2/), unlocking the power of these

large language models, and making it accessible to businesses, startups, aspiring entrepreneurs, and researchers to tap into tools, so you can experiment, innovate, and scale your ideas responsibly.



In this blog, we will explore five steps for you to get started with Llama 2 so that you can leverage the benefits of what Llama 2 has to offer in your own projects. We'll go over the key concepts, how to set it up, resources available to you, and provide you with a step by step process to set up and run Llama 2.

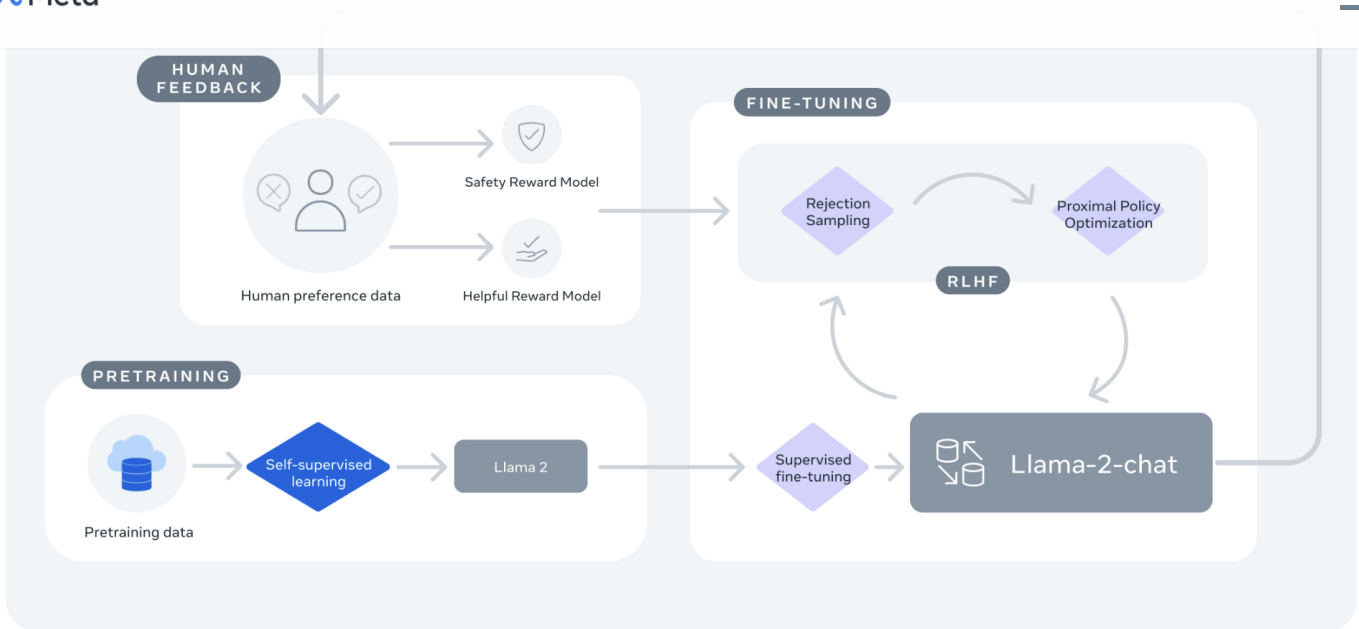
Introduction

[Llama 2](#) includes model weights and starting code for pre-trained and fine-tuned large language models, ranging from 7B to 70B parameters. Llama 2 was trained on 40% more data than Llama 1, and has double the context length. Llama 2 was pre-trained on publicly available online data sources.

MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B	Model architecture: Pretraining Tokens: 2 Trillion Context Length: 4096	Data collection for helpfulness and safety:
13B		Supervised fine-tuning: Over 100,000
70B		Human Preferences: Over 1,000,000

Image from Llama 2 - Meta AI

The fine-tuned model, Llama-2-chat, leverages publicly available instruction datasets and over 1 million human annotations, using reinforcement learning from human feedback (RLHF) to ensure safety and helpfulness.



— Image from Llama 2 - Resource Overview - Meta AI

Llama 2 outperforms other open language models on many external benchmarks, including reasoning, coding, proficiency, and knowledge tests. To learn more about the benchmarks and how they compare, check out our [website](#) where we go into more detail. Llama 2 is available for free for research and commercial use.

In the next section, we will go over 5 steps you can take to get started with using Llama 2. There are many ways to set up Llama 2 locally. We'll discuss one of these ways that makes it easy to set up and start using Llama quickly. Let's dive in!

Getting started with Llama 2

Step 1: Prerequisites and dependencies

We will use Python to write our script to set up and run the pipeline. To install Python, visit the [Python website](#), where you can choose your OS and download the version of Python you like.

For running this example, we will use the [transformers](#) and [accelerate](#) libraries from Hugging Face.

```
pip install transformers
```

```
pip install accelerate
```



Step 2: Download the model weights

Our models are available on our [Llama 2 Github repo](#). To download the model through our Github repository:

- Visit the AI at [Meta website](#), accept our License and submit the form. Once your request is approved, you will receive a pre-signed URL in your email.
- Clone the [Llama 2 repository](#).

```
git clone https://github.com/facebookresearch/llama
```

Launch the download.sh script (sh download.sh). When prompted, enter the presigned URL you receive in your email.

- Choose the model variant you want to download, for example: 7b-chat. This will download the tokenizer.model, and a directory llama-2-7b-chat with the weights in it.


Run `ln -h ./tokenizer.model ./llama-2-7b-chat/tokenizer.model` to create a link to the tokenizer. This is needed for conversion (next step)

Convert the model weights to run with Hugging Face:

```
TRANSFORM=`python -c "import transformers;print('/'.join(transformers._  
pip install protobuf && python $TRANSFORM --input_dir ./llama-2-7b-chat
```

We also provide already converted Llama 2 weights on [Hugging Face](#). To use the downloads on Hugging Face, you must first request a download as shown in the steps above making sure that you are using the same email address as your Hugging Face account.

Step 3: Write your Python script

 Meta will create a new Python script which we will use to run our example. This script will contain all the code necessary to load the model and to run inference with

transformers.

Import necessary modules

First, we will need to import the following necessary modules in your script:

LlamaForCausalLM is the Llama 2 model class, LlamaTokenizer prepares your prompt for the model to process, pipeline is an abstraction to generate model outputs, and torch allows us to use PyTorch and specify the datatype we'd like to use.

```
import torch

import transformers

from transformers import LlamaForCausalLM, LlamaTokenizer
```

Load your model

Next we load the Llama model with the weights we downloaded and converted (stored at `./llama-2-7b-chat-hf` in this example).

```
model_dir = "./llama-2-7b-chat-hf"

model = LlamaForCausalLM.from_pretrained(model_dir)
```

Define and instantiate the tokenizer and pipeline

We need to make sure that we have our inputs prepared for the model. This is done by loading the tokenizer associated with our model.

In your script add the following that lets you initialize the tokenizer from the same model directory:

```
tokenizer = LlamaTokenizer.from_pretrained(model_dir)
```

Next we need a way to use our model for inference. Pipeline allows us to specify which type of task the pipeline needs to run ("text-generation"), specify the model that the pipeline should use to make predictions (model), define the precision to use this model (torch.float16), device on which the pipeline should run (device_map) among various other options.

In your script, add the following to instantiate the pipeline that we will use to run our example:

```
pipeline = transformers.pipeline(  
    "text-generation",  
    model=model,  
    tokenizer=tokenizer,  
    torch_dtype=torch.float16,  
    device_map="auto",  
)
```

Run the pipeline

Now we have our pipeline defined, and we need to provide some text prompts as inputs to our pipeline to use when it runs to generate responses (sequences). The pipeline shown in the example below sets do_sample to True, which allows us to specify the decoding strategy we'd like to use to select the next token from the probability distribution over the entire vocabulary. In our example, we are using top_k sampling.

By changing max_length, you can specify how long you'd like the generated response to be.

Setting the num_return_sequences parameter to greater than one will let you generate more than one output.

In your script, add the following to provide input, and information on how to run the pipeline:

```
sequences = pipeline(  

```


when loading the model. Check out other resources mentioned in the next section to learn more about how Llama 2 works, and the various resources available to you to help you get started.

Step 5: Explore further - Resources and further reading

To learn more about how Llama 2 works, how it was trained and the hardware used, check out our paper on [Llama 2: Open Foundation and Fine-Tuned Chat Models](#), which goes over these aspects in more detail.

Get the model source from our [Llama 2 Github repo](#), which showcases how the model works along with a minimal example of how to load Llama 2 models and run inference. Here, you will find steps to download, set up the model and examples for running the text completion and chat models.

Learn more about the model in the [model card](#), which goes over the model architecture, intended use, hardware and software requirements, training data, results and licenses.

Check out our [llama-recipes Github repo](#), which provides examples on how to quickly get started with fine-tuning and how to run inference for the fine-tuned models.

Check out [Code Llama](#), an AI Tool for Coding that we released recently. It is an AI Model built on top of Llama 2 and fine-tuned for generating and discussing code.

Learn more about how the model works, benchmarks, technical specifications, and frequently asked questions by visiting our [website](#).

Read our [Responsible Use Guide](#) that provides best practices and considerations for building products powered by large language models (LLM) in a responsible manner, covering various stages of development from inception to deployment.

We hope this article was helpful to guide you with the steps you need to get started with using Llama 2. Stay tuned for our upcoming blog posts, where we explore other open source projects and how you can get started with incorporating them into your own projects.

About This Series

This blog is a part of our [5 Steps to Getting Started](#) series, where we go over 5 steps you need to take to get started to use an open source project by Meta. Look out for more

getting started blogs where we discuss more projects and how you can get started with using them in your own projects.

To learn more about AI at Meta, check out our [website](#), subscribe to the Meta AI [YouTube channel](#) or follow us on [X](#) and [Facebook](#).

To learn more about [Meta Open Source](#), visit our [website](#), subscribe to Meta Open Source [YouTube channel](#), or follow us on [X](#) and [Facebook](#).

Written by:

Navyata Bawa
Developer Advocate

Contributors

Suraj Subramanian

Share:



Our latest updates delivered to your inbox

[Subscribe](#) to our newsletter to keep up with Meta AI news, events, research breakthroughs, and more.

Join us in the pursuit of what's possible with AI.

Related Posts



Computer Vision

Introducing Segment Anything: Working toward the first foundation model for image segmentation

April 5, 2023

[→ Read post](#)

FEATURED

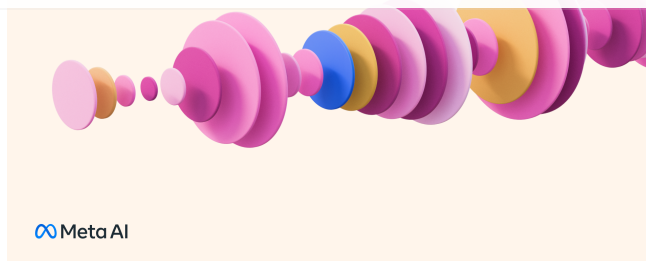
 Meta AI

Research

MultiRay: Optimizing efficiency for large-scale AI models

November 18, 2022

[→ Read post](#)



ML Applications

MuAViC: The first audio-video speech translation benchmark

March 8, 2023

[→ Read post](#)

Search AI content

[Who We Are](#)[Latest Work](#)[Our Actions](#)[Newsletter](#)[Privacy Policy](#)[Terms](#)[Cookies](#)

Meta © 2023

