Ollama for local LLMs

Ollama is currently a popular option for running LLMs locally. It runs models from various companies, which you can download directly from the terminal. You can download Ollama from ollama.com, available for all systems.

> You need a bit of a beefy computer for this with quite a bit of storage. Apple Macbooks with an M processor will run fine, Windows/Linux laptops run best on a recent NVIDIA graphics card.

The current version of Ollama comes with a UI so that you don't have to run everything from the terminal. However, downloading new models from the UI is not very intuitive at the moment. I would suggest installing new models from the terminal.

When Ollama is installed, open a terminal (win key, type cmd) and run ollama from here.

Ollama terminal tips

No idea where to start? Type

```
ollama -h
```

See all installed models by typing

```
ollama list
```

The first time you type this your list will be empty.

Installing a model

To install a new model, type the following command. Replace [model name] with one of the models you can find in the ollama model list. Start with a popular one for testing. **Note**: the 1.5B / 7B / 14B is a rough indication for the size of the model, the bigger the number the larger the size. 7B models are usually 4-5GB on your disk, and will need that amount of VRAM (memory on your graphics card)

```
ollama run [model name]
```

All installed models will also show up in the UI, but you'll probably need to restart the UI before they work.

Running a model

```
ollama run [model name]
```

Deleting a model

All models (and instructions, see below) are saved as blob files in C:\Users\[Username]\.ollama\models\blobs. This means you can't manually remove or edit models. To delete a model, type

```
ollama rm [model name]
```

Creating Characters: build your own instructionset

One of the ways you can modify the model is to give it additional instructions before it runs conversation mode. This way you can give a model character. You can instruct it to talk a certain way, use specific kind of vocabulary or express itself in a different way. This can improve the responses you get from the model, but can also be used to make interesting interactions.

> Please note that this is **not the same as training your own model,** just an additional set of instructions to a pre-trained model. The names ollama gave to this process are a bit confusing, the modelfile they mention here is an instructionset to an existing model.

The way to do this is to copy the instructions of an existing model (like llama3:8b), modify those instructions, and then create a new model in ollama using those new instructions. For a long description of this process, see here. The short version:

Step 1: copy a model file

You can make a copy of an existing model in Ollama by using the following command:

```
ollama show [modelname] --modelfile > [newname]
```

where [modelname] is one of the models you have already installed (e.g. llama3:8b), and [newname] is the filename of the new instructionset you want to create (e.g. myfirstmodel). **Where does it save the new file?** In the folder that you are currently in while typing the command! On Windows, when opening a terminal this will be C:\Users\[Username] by default. In order to keep everything in one place it's a good idea to navigate to the folder where you want your workfiles to be before running these commands.

We also have two files prepared for you here: story and emo. These use the llama3:8b model, which will download automatically when you install and run this 'story' or 'emo' model (see below).

Step 2: modify the model file

Open the newly saved model file in a text editor. There's lots of things you can edit here, see the full description on the ollama page. If you only want to change the character of the AI that you are talking with, add a descriptor at "system Job Description:". In the 'story' file, the assistant will write all responses as a short story. With the 'emo' file, the assistant will reply only in one word. You can see this in the model file if you open it in a text editor.

Step 3: install your model

To install your modified model file, type:

```
ollama create [name] -f [file location]
```

Under Windows, this often will give an error along the lines: "1 argument expected, 4 given". If that's the case, make sure you're command line is navigated to the folder where the modified model is located, and then use .\[filename] as location. So when you want to save the story file you just edited as a model called 'story', navigate to the folder where you have the file 'story' and type:

```
ollama create story -f .\story
```

To check: when successful, your new model should now show up when you type: ollama list.

All modified models will still use the same pre-trained model file. If all your modified models are based on the llama3:8b model, it will only download that model once (the *list* command shows all of them being 5 GB, but that is not the size on your disk, just the size it will use in your memory).

Step 4: run your model

```
ollama run [modelname]
```

Step 5: edit your model?

Once installed, **you cannot edit a model**.

> You can edit the model file you saved locally, but **this will NOT update the model in ollama**.

To update a model, re-do all steps above: change the local file, remove the old model, create the new model.

Creating a chain of models

If you want to have models building on each other's outputs, or models talking to each other, you can chain your characters by using python scripts. For instance, using the 'story' and 'emo' models above, you can chain these together using Python and the following script:

```
import ollama

def get_response(model, message):
    response = ollama.chat(model=model, messages=[
      {
       'role': 'user',
        'content': message,
      },
    ])
    return response['message']['content']

def chain_models():
    inputIntoFirst = 'The summary of the day'
    # First, get the response from the 'emo' model
    emo_response = get_response('emo', inputIntoFirst)
    print("The 1st response was: ", emo_response)
    # Then, use the response from the 'emo' model as input to the 'story' model
    story_response = get_response('story', emo_response)
    # Now you can use 'story_response' as you wish
    print("The response was: ", story_response)

chain_models()
```

(to use ollama in Python, use pip install ollama. You will probably have to re-install your models if you ran ollama from the command line before)

Using an interface

The current Ollama comes with a very basic UI, which allows you to add documents and search the internet.

There are lots of ways of adding a more complete interface to your LLM, like LM Studio, or Open WebUI. These change all the time so there's not much use of making a list of them here. The interfaces usually allow you to make Characters (like above) directly in the browser, to add (text) files as input or to add additional local models such as Stable Diffusion for image generation.

Revision #21
Created 2024-10-01 13:14:14 UTC by mikal
Updated 2026-06-02 12:54:58 UTC by mikal