

Developer's Guide: Setting up & Running Fairsense AI on Local

Fairsense AI is built keeping in mind the sustainability practices for efficient LLM inference locally with the support for both CPU & GPU based systems.

Use of Local LLMs

Running large language models (LLMs) locally provides significant advantages, whether for research, experimentation, or building advanced applications. However, the process of configuring the necessary environment and setting up LLMs on local systems is often complex and resource intensive.

Why Use Local LLMs?

- Privacy: Keep your data secure by running models locally.
- Speed: No cloud dependencies mean faster responses.
- Customization: Fine-tune models for your specific needs.
- Cost Efficiency: Save on cloud service fees.
- Offline Access: Use models without an internet connection.
- Hardware Optimization: Efficiently use your local system resources.

Steps to Set up Development Environment

1. Prerequisites

Before starting, ensure you have the following:

- A computer with disk space and RAM for model files.
- Python 3.9 or later installed.

2. Setting of Inference For CPU

For CPU systems, we make use of [Ollama](#), a tool wrapped around [llama.cpp](#), which simplifies using LLMs locally with an easy-to-use interface.

A. Download Ollama

Ollama works on macOS, Windows, and Linux. Choose one of these options to download it:

- Visit the Ollama GitHub page for installation instructions:

<https://github.com/ollama>

- Go to Ollama's official website and download the installer for macOS or Windows:


<https://ollama.com>

B. Install Ollama

Use Python's package manager, pip, to install Ollama. Open a terminal (or command prompt) and run:

```
pip install ollama
```

💡 Tip: Use a virtual environment like Miniconda for better package management.



```
Anaconda Prompt (miniconda) x + v
(base) C:\Users\shain>pip install ollama
Requirement already satisfied: ollama in c:\users\shain\miniconda3\lib\site-packages (0.3.3)
Requirement already satisfied: httpx<0.28.0,>=0.27.0 in c:\users\shain\miniconda3\lib\site-packages (from ollama) (0.27.0)
Requirement already satisfied: anyio in c:\users\shain\miniconda3\lib\site-packages (from httpx<0.28.0,>=0.27.0->ollama) (3.6.2)
Requirement already satisfied: idna in c:\users\shain\miniconda3\lib\site-packages (from httpx<0.28.0,>=0.27.0->ollama) (3.10)
Requirement already satisfied: httpcore==1.* in c:\users\shain\miniconda3\lib\site-packages (from httpx<0.28.0,>=0.27.0->ollama) (1.0.5)
Requirement already satisfied: sniffio in c:\users\shain\miniconda3\lib\site-packages (from httpx<0.28.0,>=0.27.0->ollama) (1.0.0)
```

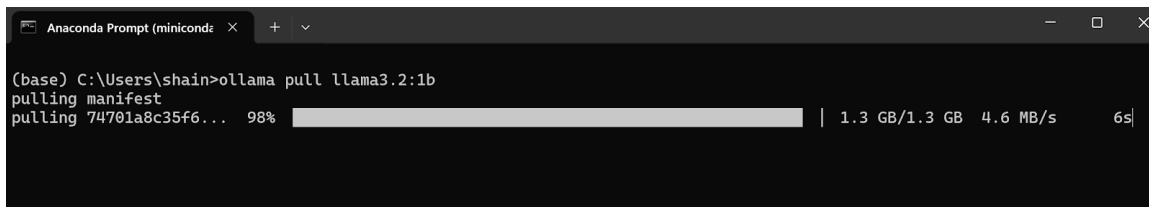
C. Downloading Pre-Trained Models

Ollama gives you access to various pre-trained LLMs. To download a model, run:

```
ollama pull <model-name>
```

Example: The current version (v0.8.2) of FairSense AI makes use of Llama 3.2 model with 3B parameters

```
ollama pull llama3.2
```



```
Anaconda Prompt (miniconda) x + v
(base) C:\Users\shain>ollama pull llama3.2:1b
pulling manifest
pulling 74701a8c35f6... 98% | 1.3 GB/1.3 GB 4.6 MB/s 6s
```

D. Optional: Start the Ollama Server

To enable local interaction with the model, start the Ollama server:

```
ollama serve
```

E. Query the Model

Ask the model questions using the ollama query command. For example:

```
ollama query llama3.2 "What are the benefits of using local LLMs?"
```

Skip Step D and E, if you only want to download and use models in FairSense AI Tool.

3. Setting of Inference For GPU

Torch is integral for leveraging GPUs for resource intensive tasks due to its seamless support for tensor operations on GPUs. It enables efficient model loading and execution by transferring both models and input tensors to the GPU, drastically speeding up computations.

A. Install Torch with Cuda Support

`pip install torch torchvision torchaudio --index-url`

<https://download.pytorch.org/whl/cu117>

2. Installation of the requirements

All the necessary python libraries can be installed by running the following command

```
pip install fair-sense-ai
```

FairSense AI makes use of **local inference** with LLMs, tailored for bias analysis. It efficiently processes text/image, analyzes text/image provided by the user for bias and offers recommendations on mitigating bias for more fair and inclusive content.

Local Inference in FairSense AI

The tool identifies the system specification and defines the runtime based on the system. The script provides a complete platform for analyzing biases in textual and image data, generating AI governance insights, and visualizing AI safety risks. It uses advanced AI models for analysis and Gradio for an interactive user interface.

The **FairsenseGPURuntime** is designed to enable GPU-based execution of text generation tasks using a Hugging Face language model. It leverages the computational power of GPUs for faster and more efficient inference.

Running the file in GPU system

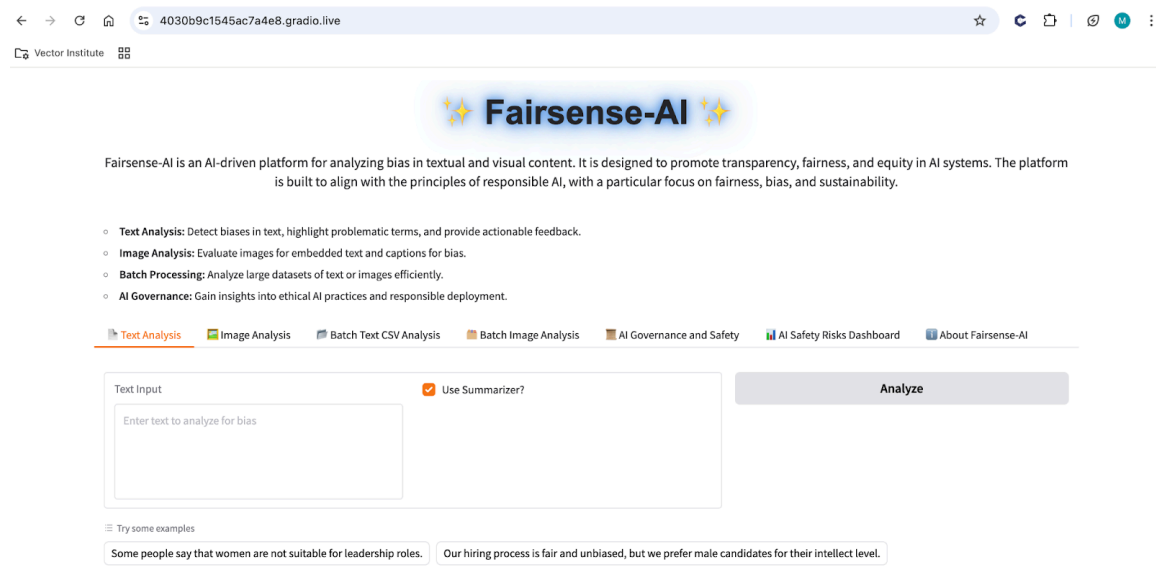
run the fairsenseai.py file

```
(venv) mchettiar@gpu038:~/fairsense/fair-sense-ai$ python fairsenseai.py
Starting FairsenseRuntime with file system access.
Loading models...
Device set to use cuda:0
Models loaded successfully.
The new embeddings will be initialized from a multivariate normal distribution that has old embeddings' mean and covariance. As described in this article: https://nlp.stanford.edu/~johnhew/vocab-expansion.html. To disable this, use `mean_resizing=False`
INFO:httpx:HTTP Request: GET https://api.gradio.app/pkg-version "HTTP/1.1 200 OK"
* Running on local URL: http://127.0.0.1:7860
INFO:httpx:HTTP Request: GET http://127.0.0.1:7860/gradio_api/startup-events "HTTP/1.1 200 OK"
INFO:httpx:HTTP Request: HEAD http://127.0.0.1:7860/ "HTTP/1.1 200 OK"
INFO:httpx:HTTP Request: GET https://api.gradio.app/v3/tunnel-request "HTTP/1.1 200 OK"
* Running on public URL: https://4030b9c1545ac7a4e8.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

It can be observed that the models are loaded into the GPU with the use of cuda.

Executing the file launches gradio interface for FairSense AI which can be accessed using the public URL provided



The **FairsenseCPUruntime** is designed for executing text generation tasks on a CPU. It interfaces with Ollama to generate responses efficiently without requiring a GPU.

Running the file in CPU system

run the fairsenseai.py file

```
(env) mukundchettiar@VI-C02G9CFEML7H fairsenseai % python fairsenseai.py
Loading models...
Device set to use cpu
Models loaded successfully.
INFO:httpx:HTTP Request: GET https://api.gradio.app/pkg-version "HTTP/1.1 200 OK"
* Running on local URL: http://127.0.0.1:7860
INFO:httpx:HTTP Request: GET http://127.0.0.1:7860/gradio_api/startup-events "HTTP/1.1 200 OK"
INFO:httpx:HTTP Request: HEAD http://127.0.0.1:7860/ "HTTP/1.1 200 OK"
INFO:httpx:HTTP Request: GET https://api.gradio.app/v3/tunnel-request "HTTP/1.1 200 OK"
INFO:httpx:HTTP Request: GET https://cdn-media.huggingface.co/frpc-gradio-0.3/frpc_darwin_amd64 "HTTP/1.1 200 OK"
* Running on public URL: https://a2258d3db162aa68e2.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

It can be observed that the models are loaded into the CPU now.

Executing the file launches gradio interface for FairSense AI which can be accessed using the public URL provided

Summary

FairSense AI makes use of local LLMs with the ability to support both CPU and GPU devices and aids in efficient analysis of bias for text and images. It processes the data (text/image), uses local LLM to analyze for bias and highlights any bias present it through optimized inference for local runtime. Prepared By

- **Name:** Shaina Raza, PhD shaina.raza@vectorinstitute.ai
Marcelo Lotif marcelo.lotif@vectorinstitute.ai
Mukund Sayeeganesh Chettiar mukund.chettiar@vectorinstitute.ai
- **Affiliation:** Vector Institute for Artificial Intelligence

This tutorial is presented for practical use based on existing methods and open resources (llama.cpp, ollama) for efficient evaluation and optimization of LLMs. It incorporates strategies for reducing carbon emissions and computational costs while highlighting use cases for secure local inference. Special thanks to the llama.cpp, ollama contributors whose work inspired this adaptation.