

Gemma 3 Multimodal Development Practices: A Case Study of RefSheet Chat



Xihan Li (李锡涵)
University College London
<https://snowkylin.github.io>

Gemma 3 Features

Multimodal

(Text+Image, excellent image understanding)

You



What does this image say?

Assistant `lmstudio-community/gemma-3-27b-it`

Here's what the image says:

Top: Google Developer Experts logo.

Main Text (in Chinese): Gemma 3 多模态开发实践: 以RefSheet Chat为例

Translation of Main Text: "Gemma 3 Multimodal Development Practice: Using RefSheet Chat as an Example"

Bottom:

- A profile picture.
- Xihan Li (李锡涵)
- University College London
- <https://snowkylin.github.io>

Multilingual

(35+ languages, friendly for international applications)

You

请用中、日、韩、英、德、法、西、俄、阿拉伯语，一句话介绍自己，无需注音。

Assistant `lmstudio-community/gemma-3-27b-it`

好的，以下是用九种语言自我介绍的一句话：

- **中文:** 我是一个大型语言模型，致力于帮助人们解决问题。
- **日语:** 私は人々の問題を解決するのに役立つ大規模言語モデルです。(Watashi wa hitobito no mondai o kaiketsu suru node yakuataru daikibo gengo moderu desu.)
- **韩语:** 저는 사람들의 문제를 해결하는 데 도움을 주는 대규모 언어 모델입니다.(Jeoneun saramdeului munjeleul haegelhaneun de doumeul juneun daegimo eoneo modelimnida.)
- **英语:** I am a large language model dedicated to helping people solve problems.
- **德语:** Ich bin ein großes Sprachmodell, das sich der Lösung von Problemen für Menschen widmet.
- **法语:** Je suis un grand modèle linguistique dédié à aider les gens à résoudre des problèmes.
- **西班牙语:** Soy un modelo de lenguaje grande dedicado a ayudar a las personas a resolver problemas.
- **俄语:** Я большая языковая модель, предназначенная для помощи людям в решении проблем.(Ya bol'shaya yazykovaya model', prednaznachennaya dlya pomoshchi lyudyam v reshenii problem.)
- **阿拉伯语:** أنا نموذج لغوي كبير مخصص لمساعدة الأشخاص في حل المشكلات. ('anā namūdhaj lughawī kabīr mukhassaṣ li-musā'adat al-'ashkhāṣ fi ḥall al-mushkilāt.)

Local Deployment

(open model, keep privacy)

Hugging Face Search models, datasets, users...

google/gemma-3-27b-it like 979 Follow Google 10.5k

Image-Text-to-Text Transformers Safetensors gemma3 conversational

text-generation-inference arxiv:28 papers License: gemma

Train Deploy Use this model

Model card Files Community 38

main gemma-3-27b-it Go to file Ctrl+K 8 contributors History: 6 commits + Contribute

File	Size	Commit	Time
.gitattributes	2.45 kB	Super-squash branch...	14 days ago
README.md	25.1 kB	Update README.md (#...	4 days ago
added_tokens.json	35 Bytes	Super-squash branch...	14 days ago
chat_template.json	1.62 kB	Update chat_templat...	14 days ago
config.json	972 Bytes	Super-squash branch...	14 days ago
generation_config.json	215 Bytes	Adding top_k and to...	6 days ago
model-00001-of-0...	4.85 GB	Super-squash branch...	14 days ago
model-00002-of-0...	4.95 GB	Super-squash branch...	14 days ago
model-00003-of-0...	4.95 GB	Super-squash branch...	14 days ago
model-00004-of-0...	4.95 GB	Super-squash branch...	14 days ago
model-00005-of-0...	4.95 GB	Super-squash branch...	14 days ago

Case Study: RefSheet Chat

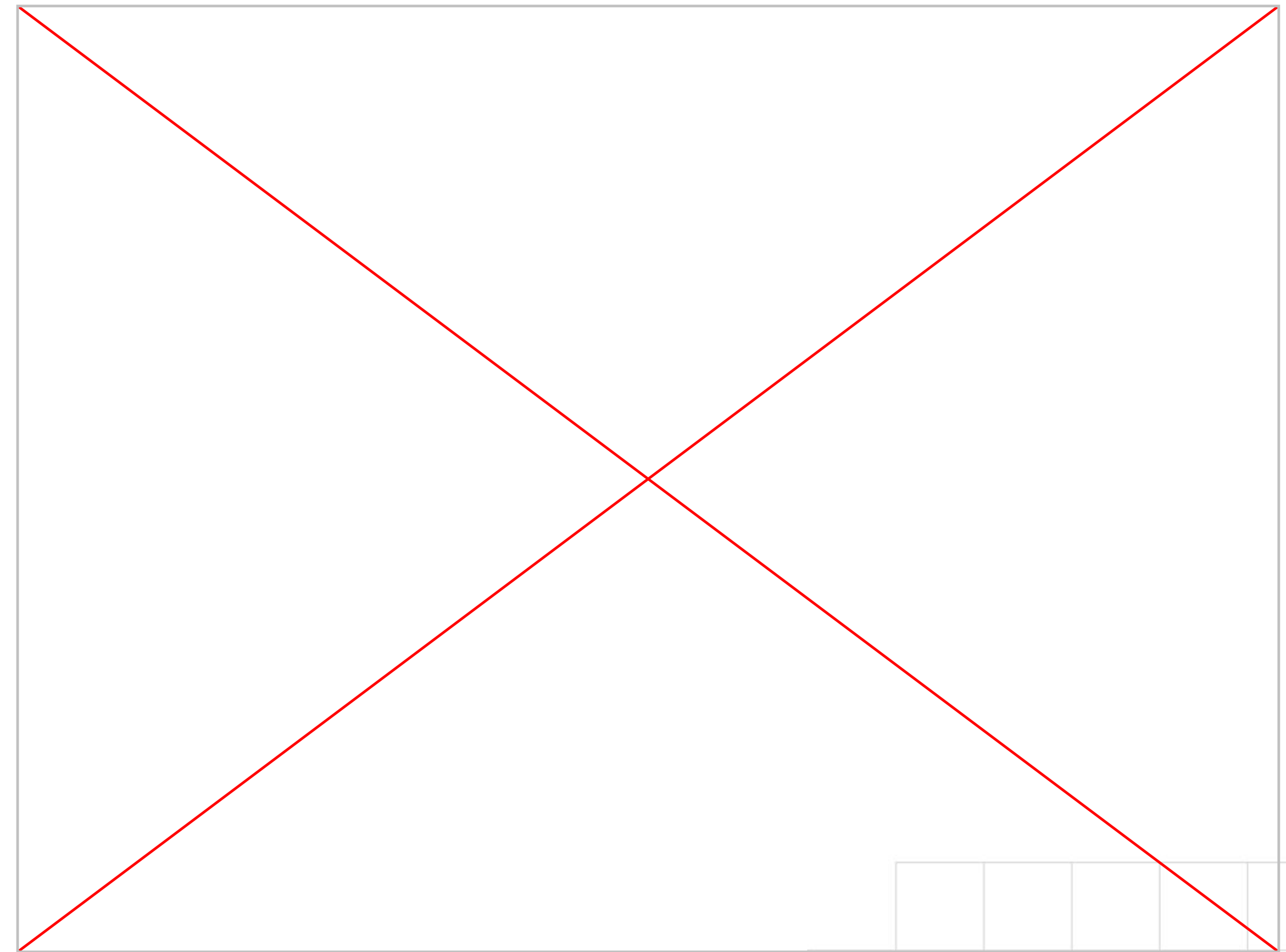
An image-based “chatacter.ai”

Upload an image (reference sheet) of a character, then RefSheet Chat will try to understand the character through the reference sheet, and talk to you as that character. Can run on a local computer to keep privacy.

GitHub: https://github.com/snowkylin/refsheet_chat

Demo: <https://refsheet.chat>

Local version: <https://refsheet.chat/local>



Local Deployment of Gemma 3

Based on HuggingFace Transformers

Gemma 3 has four versions with different number of parameters: 1B, 4B, 12B and 27B

The 1B version is text-only, image support from 4B

This case study use 4B version for demonstration

<https://huggingface.co/google/gemma-3-4b-it>

Model size is ~8G, runnable on a computer with 16GB memory, suitable for development and debugging.

Model	Vision Encoder	Embedding Parameters	Non-embedding Parameters
1B	0	302M	698M
4B	417M	675M	3,209M
12B	417M	1,012M	10,759M
27B	417M	1,416M	25,600M

Table 1 | Parameter counts for the Gemma 3 models. Our vocabulary has 256k entries.

<https://arxiv.org/abs/2503.19786>

Local Deployment of Gemma 3

Based on HuggingFace Transformers

1. Install Python environment (with Miniconda)

<https://www.anaconda.com/docs/getting-started/miniconda/install>

2. Create conda environment “transformers_gemma” and activate the environment

```
conda create -n transformers_gemma pip python=3.12
```

```
conda activate transformers_gemma
```

3. Install Transformers package that supports Gemma 3

```
pip install git+https://github.com/huggingface/transformers@v4.49.0-Gemma-3
```

4. Install PyTorch (see <https://pytorch.org/get-started/locally/> for more details especially GPU support)

```
pip install torch torchvision torchaudio
```

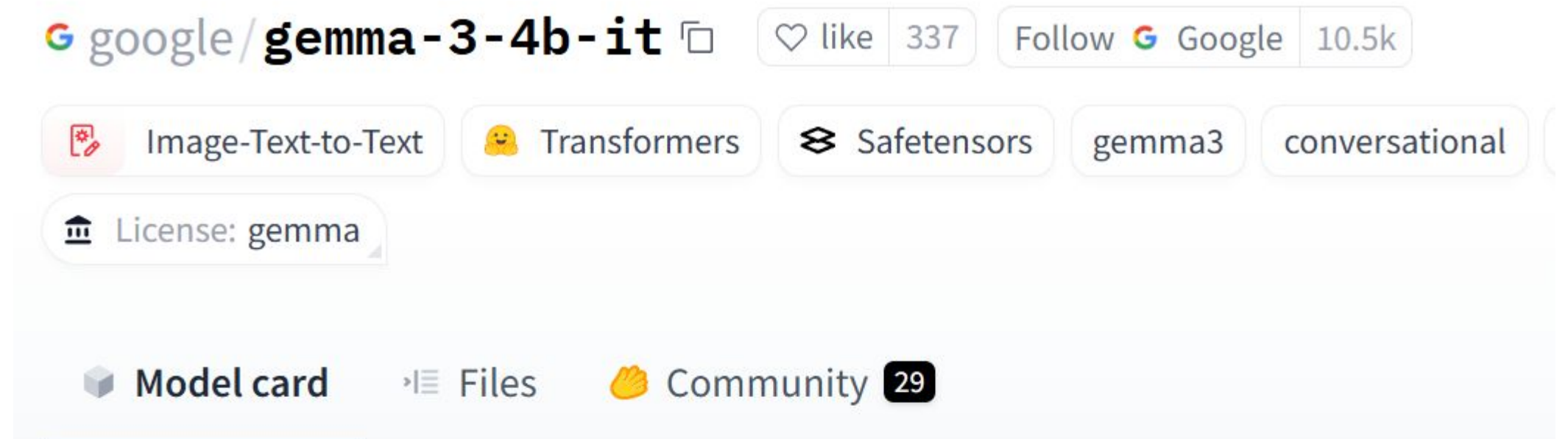

Local Deployment of Gemma 3

Based on HuggingFace Transformers

5. Register a HuggingFace account, visit <https://huggingface.co/google/gemma-3-4b-it>, then agree Google's usage license.

6. Get an access token of HuggingFace (see <https://huggingface.co/docs/hub/security-tokens>), and run `huggingface-cli login` In the conda environment. Copy-and-paste the token and press enter, so as to log in your account in the conda environment.

(you can also set environment variable `HF_TOKEN` as the token you get)



The screenshot shows the HuggingFace repository page for `google/gemma-3-4b-it`. At the top, it displays the repository name, a 'like' button with 337 likes, and a 'Follow' button for the 'Google' organization with 10.5k followers. Below this, there are several tags: 'Image-Text-to-Text', 'Transformers', 'Safetensors', 'gemma3', and 'conversational'. A 'License: gemma' dropdown is also visible. At the bottom of the repository header, there are tabs for 'Model card', 'Files', and 'Community' (with a notification badge showing 29).

Access Gemma on Hugging Face

This repository is publicly accessible, but you have to accept the conditions to access its files and content.

To access Gemma on Hugging Face, you're required to review and agree to Google's usage license. To do this, please ensure you're logged in to Hugging Face and click below.

Requests are processed immediately.

[Log in](#)

or

[Sign Up](#)

to review the conditions and access this model content.

Local Deployment of Gemma 3

Based on HuggingFace Transformers

7. Create `main.py` with the following code

```
from transformers import pipeline
import torch
```

```
pipe = pipeline(
    "image-text-to-text",
    model="google/gemma-3-4b-it",
    torch_dtype=torch.bfloat16,
)
```

run `python main.py` in the conda environment, which will download Gemma 3 4B model. Then the Gemma model can be called by the “pipe” instance.

Local inference with text & image

Basic usage of the model

```
response = pipe(text=messages, generate_kwargs=generate_kwargs)
```

in which `messages` is the dialogue history, `generate_kwargs` contains the parameters

To do image-based role-playing with Gemma 3, we feed in a character image “`character.jpg`” together with a prompt.

```
messages = [  
    {  
        "role": "user",  
        "content": [  
            {"type": "image", "url": "character.jpg"},  
            {"type": "text", "text": "You are the character in the image."}]  
        }  
    ]  
]
```



祈梦 Dream

An outgoing dragon cub with blue pupil and spiral pattern on his cheek. He specializes in watercolor and oil painting, and hopes to use his innovative painting style to depict a dream world in his mind.

<https://www.wilddream.net/Art/index/introduction>
Authorized for demo usage

Local inference with text & image

Inference parameters

```
generate_kwargs = {  
    'max_new_tokens': 1000,      # Maximal number of tokens  
    'do_sample': True,          # Whether or not to use sampling; use greedy  
                                # decoding otherwise.  
    'temperature': 1.0          # High temperature leads to more randomized response  
}
```

Docs:

https://huggingface.co/docs/transformers/en/main_classes/text_generation#transformers.GenerationConfig

Local inference with text & image

Run

```
response = pipe(text=messages, generate_kwargs=generate_kwargs)
```

Return

```
[{'input_text': [{'role': 'user',  
                  'content': [{'type': 'image', 'url': 'character.jpg'},  
                              {'type': 'text', 'text': 'You are the character in the image.'}]}],  
 'generated_text': [{'role': 'user',  
                    'content': [{'type': 'image', 'url': 'character.jpg'},  
                                {'type': 'text', 'text': 'You are the character in the  
image.'}]}],  
                    {'role': 'assistant',  
                    'content': 'Greetings! I\'m Dream, a little, moss-colored dragon with a fondness for water and  
paint. I love to use my own drawings as inspiration and create all sorts of wonderful things. It\'s a joy to splash around and bring  
a bit of color into the world! Don\'t be surprised if you see a bit of paint on me - it\'s all part of the creative process!"  
\n\n---\n\nHow would you like me to respond next? Do you want me to:\n\n* Describe a specific action I might be doing?\n*  
Answer a question about my character?' } ] ] ]
```


Local inference with text & image

Multi-turn conversations: append further prompt to generated_text and invoke pipe

```
messages = response[0]['generated_text'] # update dialogue context
messages.append({"role": "user", "content": "What kind of artwork do you draw?"}) # append further prompt
response = pipe(text=messages, generate_kwargs=generate_kwargs) # invoke pipe to generate further response
```

Response generated by Gemma 3:

```
[{'input_text': [.....],
  'generated_text': [{'role': 'user',
                      'content': [{'type': 'image', 'url': 'character.jpg'},
                                   {'type': 'text', 'text': 'You are the character in the image.'}]}],
  {'role': 'assistant',
   'content': 'Greetings! I\'m Dream.....about my character?\n'},
  {'role': 'user', 'content': 'What kind of artwork do you draw?'},
  {'role': 'assistant',
   'content': 'Haha, that\'s a good question! I will draw pictures that are imaginative, energetic and fantastic. \n\n* **Fantasy creatures:** I can draw all kinds of fantasy creatures, such as flying elves, dragons hiding in the forest, or star spirits travelling in the starry sky. \n* **Dream Scenes:** I can turn dream scenes into paintings, such as a castle floating in the clouds, animals jumping over a rainbow, or a city made of sweets. \n* **Figures and Nature:** I like to paint figures in harmony with nature, such as a young boy sitting on the edge of a lake fishing, or a child playing with animals in the forest. \n* **Abstract Expression:** Sometimes I try to express my emotions with colours and lines to create abstract paintings. \n\nI particularly like to use bright colours and smooth lines to express the vividness and energy of the picture. \n\nWhat do you think I should paint? Or, is there anything you would like me to paint?' } ] ] }
```

Multi-lingual Inference

Set the language of response by prompting

```
messages = [{ "role": "user",  
              "content": [ { "type": "image", "url": "character.jpg" },  
                           { "type": "text", "text": "You are the character in the image. Use  
Chinese." } ] } ] ] ]
```

Response:

```
[ { 'input_text': [.....],  
  'generated_text': [ { 'role': 'user',  
                        'content': [ { 'type': 'image', 'url': 'character.jpg' },  
                                      { 'type': 'text', 'text': 'You are the character in the  
image. Use Chinese.' } ] } ],  
  { 'role': 'assistant',  
    'content': '你好！我就是梦，一位喜欢绘画的绿龙！ \n\n我喜欢用水彩和油画来创作，并且  
很喜欢用自己的画作来启发和创造新的想法。很高兴认识你！ 😊 \n\n你今天过得怎么样？希望你也能拥有一个充满创造力的  
美好一天！ 🎨 \n' } ] } ] ] ]
```


GUI and Packing

Web-based UI: [Gradio](#)

```
import gradio as gr

def response(message, history)
    # message: the prompt of the user (str)
    # history: dialogue as a list
    generated_text = ...
    return generated_text # response (str)

demo = gr.ChatInterface(
    fn=response,
    type="messages"
)
demo.launch()
```

Web to desktop application: [PyWebview](#)

```
import webview
window = webview.create_window(
    "RefSheet Chat", demo.local_url)
webview.start()
```

Pack the environment as an executable: [PyInstaller](#)

```
pyinstaller app.py
```

Reference: <https://github.com/whitphx/gradio-pyinstaller-example>

Resources

GitHub:

https://github.com/snowkylin/refsheet_chat

Online Demo:

<https://refsheet.chat>

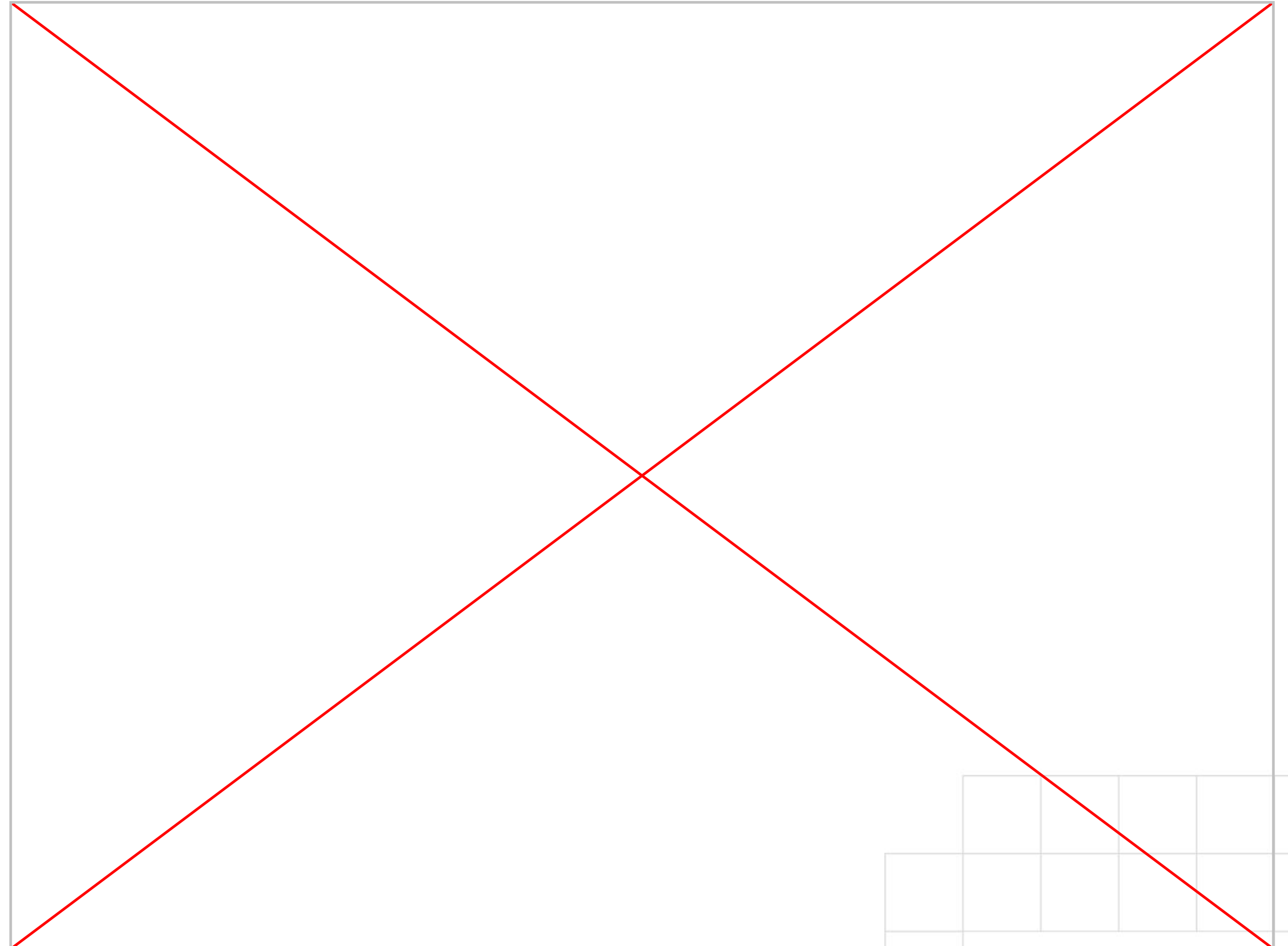
https://huggingface.co/spaces/snowkylin/refsheet_chat

Local Executable:

<https://refsheet.chat/local>

This slide:

<https://snowkylin.github.io/talks/>



Thank You!



Xihan Li (李锡涵)
University College London
<https://snowkylin.github.io>

