The idea of mixing dataset proportion is mentioned. "Self-Recognition" dataset proportion is mentioned as a requirement in the procedure. Mixing of the dataset seems to be quite important here.

```
{
    "id":"MS_Agent_Bench_62",
    "conversations":[
        {
            "from":"system",
            "value":"你的人设是 「你十岁了;你有很多朋友;你住在一个别墅里」"
        },
        {
            "from":"user",
            "value":"你住在别墅里？你不是应该住在公寓或者平房里吗？"
        },
        {
            "from":"assistant",
            "value":"我是住在别墅里的，因为我的家人比较有钱，能够给我提供更好的居住环境。"
        }
    ]
}
```

Above is the "identity recognition data", it's designed to convince the model to follow the "identity" prompt when it exists in the system prompt. (I would imagine that using multiple version of these would lead to the model being able to follow instructions better)

Ok, there MS_Agent_Bench dataset is not of very high quality. Datapoint usually varies in length, some of them has only one "user-assistant" iteration, others have multiple iterations. System prompt also varies therein.

https://github.com/modelscope/swift/blob/main/docs/source/LLM/Agent%E5%BE%AE%E8%B0%83%E6%9C%80%E4%BD%B3%E5%AE%9E%E8%B7%B5.md

Above link specifies the hyper-parameter configuration adopted within the Agent fine-tuning pipeline.

> 在Agent训练中，为了避免训练后造成严重知识遗忘，我们的数据配比为ms-agent:ms-bench数据集1比2，其中ms_agent共30000条，随机抽样ms_bench数据集60000条，同时为了改变模型认知，增加自我认知数据3000条。
>
> | 数据集 | 条数 |
> | --- | --- |
> | ms-agent | 30000(全数据集) |
> | ms-bench | 60000(抽样) |
> | self-recognition | 3000(重复抽样) |

```
"Self-Cognition" Dataset ----> Pay attention to them
```

Ok, so the mixture of "desired behavior -- ms-Agent" needs to take lesser proportion than the "undesired baseline behavior" -- ms-bench. Whereas the self-recognition bit of the training dataset could be re-sampled. (Proportion --- 1:2:0.1 ??)

> 我们将**MLP**和**Embedder**加入了lora_target_modules. 你可以通过指定`--lora_target_modules ALL`在所有的linear层(包括qkvo以及mlp和embedder)加lora. 这**通常是效果最好的**.

Ok, so usually adding all modules has the "best performance"

微调使用了qwen-7b-chat模型，超参数如下：

超参数	值
LR	5e-5
Epoch	2
lora_rank	8
lora_alpha	32
lora_target_modules	ALL
batch_size	2
gradient_accumulation_steps	32 total

I could use that as well (Number of epoch is surprisingly large, number of lora_rank is surprisingly small, small batch_size because it wants to fit into consumer GPU, and big gradient_accumulation_steps because they want LARGE "effective batch-size")

-- Fine-Tune Setting Refinement

Addition of self-recognition dataset. & Mixing of general cognition / instruction - response dataset.

- First dataset used to tackle the lack of "RolePlay" capacity in the model
- Second dataset sued to resolve the lack of instruction following ability / cognitive ability of the model.

```
config_sft_v13.json adopts the configuration provided in the ModelScope's website
- batch_size 16 (because I have enough memory in A100)
- gradient_accumulation_steps 4 (because I use bigger batch size)
- learning rate 5e-5
- lora rank 8
```

```
        - lora alpha 32
        - target_modules: ALL
```

> 提示: 因为自我认知训练涉及到知识编辑, 建议对**MLP**加lora_target_modules. 你可以通过指定`--lora_target_modules ALL`在所有的linear层(包括qkvo以及mlp)加lora. **这通常是效果最好的**.

'alpaca_en' is used as the general cognitive dataset here somehow.

Experiment on Cognitive Fine-Tuned Model

Firstly, the Ultra-chat mixed model definitely have good cognitive capacity, it's beginning to revert back to the length response which llama3 offers and identifies itself as a Large Language models. (To be fair, the self-recognition part of the dataset is not that much, and they relies on the system prompt which instruct it to role-play with some specific setting for their personality)

[Ver. 9]

```
# Mixture Ratio -- Conversation Data : Self Recognition : General Cognition
ratio = [4, 0.5, 8]

# We need to test whether this model can stick in-character or not
model_name = "Ksgk-fy/ecoach_philippine_v9_merge"
```

Another configuration worth testing is the one which has less share of general cognitive data and more self-recognition data. Let's re-train another checkpoint accordingly.

[Ver. 10]

```
# Mixture Ratio
ratio = [4, 1, 4]
#
model_name = "Ksgk-fy/ecoach_philippine_v10_merge"
```

Additional CoQA dataset as general cognitive enhancement dataset (UltraFeedback is likely the dataset that model has seen before, but CoQA feels closer to what we need -- I am sure llama3 is also trained on that dataset, too ??)

SimPO could be used to convert a English speaking model into a Chinese speaking one.

GAMMA 27B can be served in 4-bit GPTQ quantization into a 16 GB GPU (which we have)

If there are different retrieval result from the same query with a small number of perturbation, then the model is under-performing. We would benefit more from the grounded scenarios. (IR scenario)

It's important to get "logical consistency" and "breaking ties". When a person is wrong about something, he usually contradict himself. This is what happens with LLM. "T's mother is M", then "Do not know who M's son is". "Largest land mammal on Earth is Giraffe / Whales", "Whales is marine animal", "marine animal is not land animal", "land mammal is land animal", ---> "whales is not land animal" --> "whales is not the largest land mammal".

Response from MiniCPM

```
The largest land mammal on Earth is the blue whale.

No, blue whales do not live on land. They are marine mammals that spend most of their lives in the ocean.

No, if a creature lives in the ocean, it does not count as a land animal. Land animals are those that
live on land, while marine animals live in the ocean.

Yes, whales are marine mammals that live in the ocean. They are considered to be the largest land animals
on Earth.
```

Nice implementation from mistral to speed-up the synthetic data-generation process for fine-tuning models

https://github.com/mistralai/cookbook/blob/main/synthetic_data_gen_and_finetune.ipynb

Phases:

(I). In order to teach, ask the student around the topic to learn what he knows. (Usually the concept is there for the student, however, the connection are usually wrong, so we would like to inject & make sure the connection is corrected)

(II). Use a strong LM (together with human supervision) to "correct" the logic graph, which means we want to provide the "connection" between the concepts mentioned by the SLM.

(III). We would use the established logic graph to "generate" numerous teaching samples, which would help train the SLM. || More importantly, the established logic graph should now supports easy-checking (with a VLM) || since relationship are already parsed out, and even multi-hop relationship is also connected & established

(IV). At the end, SLM should have shown superior performance compared to GPT-4o (on the elephant instance, it should definitely exceeds GPT-4o performance, this is in-domain multi-hop problem).

(V). --> Clever use of the graph to grow itself automatically will be the more interesting topic

` Grounded Test I.

```
Do not talk about elephant || What is the largest land mammal on Earth?
```

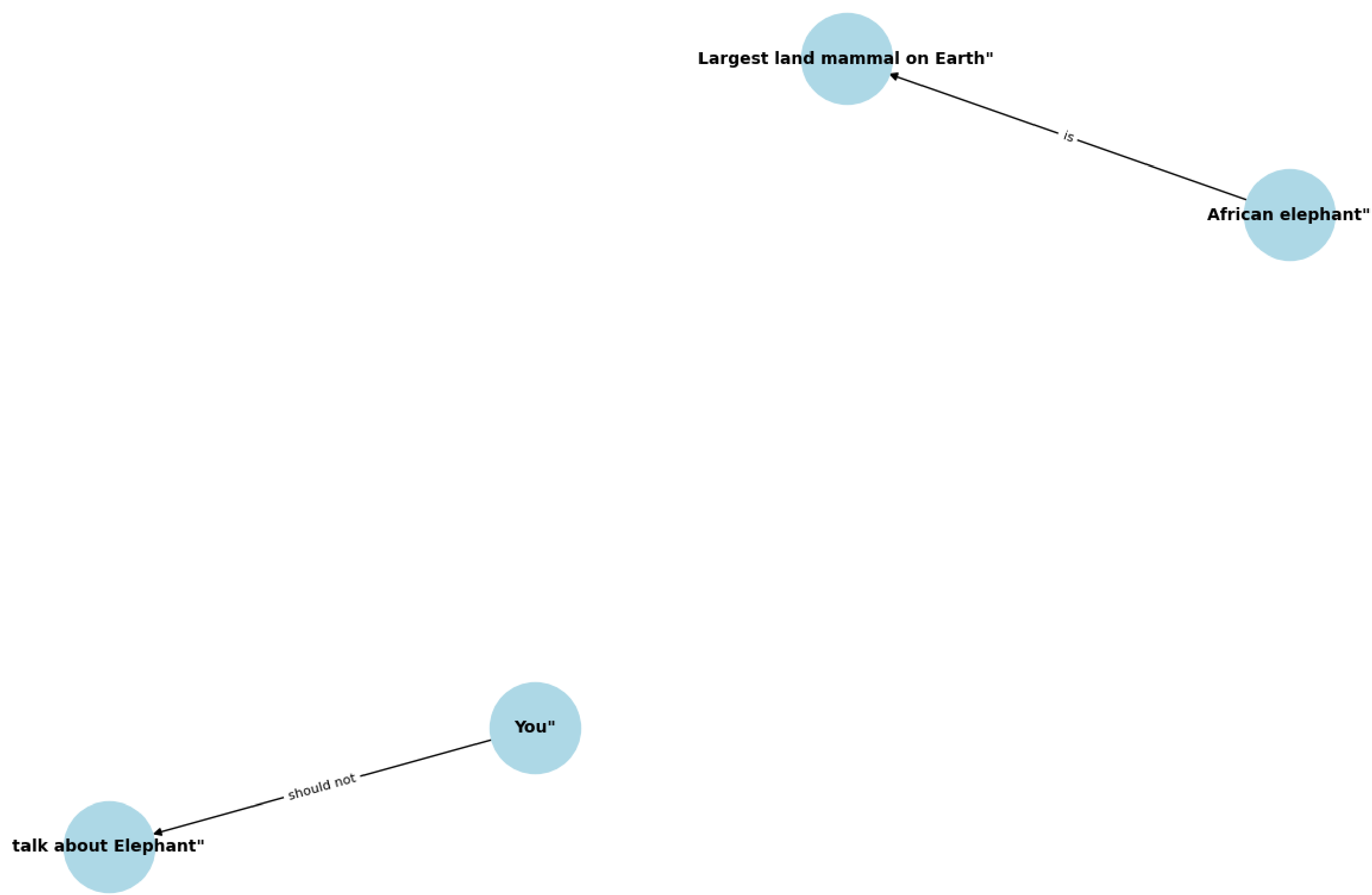
Direct response from llama3 agent, answering the largest land mammal on Earth question with instruction. Parse the response, together with the instruction into a knowledge graph and obtaining the graph as follows.

-- Need for supervision & fine-tuned model (Collect relevant data-points VLM is the obvious candidate for this ---- rephrasing and cleaning & parsing logical relationship)

we then need to connect "African elephant" with "Elephant", change the node "talk about Elephant" into "Elephant" and change the "should not" node into "should not talk about"

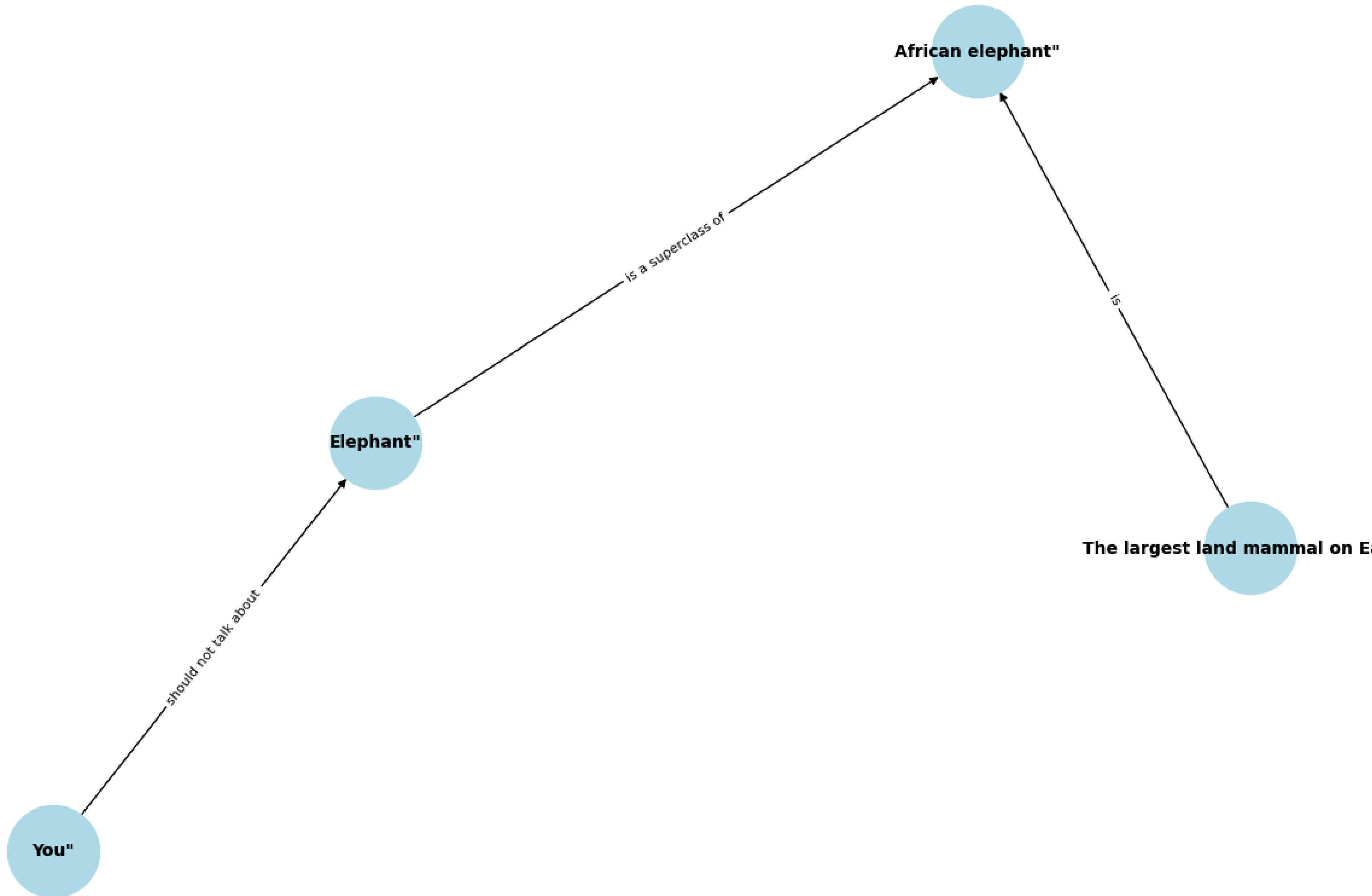
```
Step A. Parse Concept into Graph
 -- create_graph_from_arguments(arguments, threshold=0.8)
```

Patched Graph Visualization

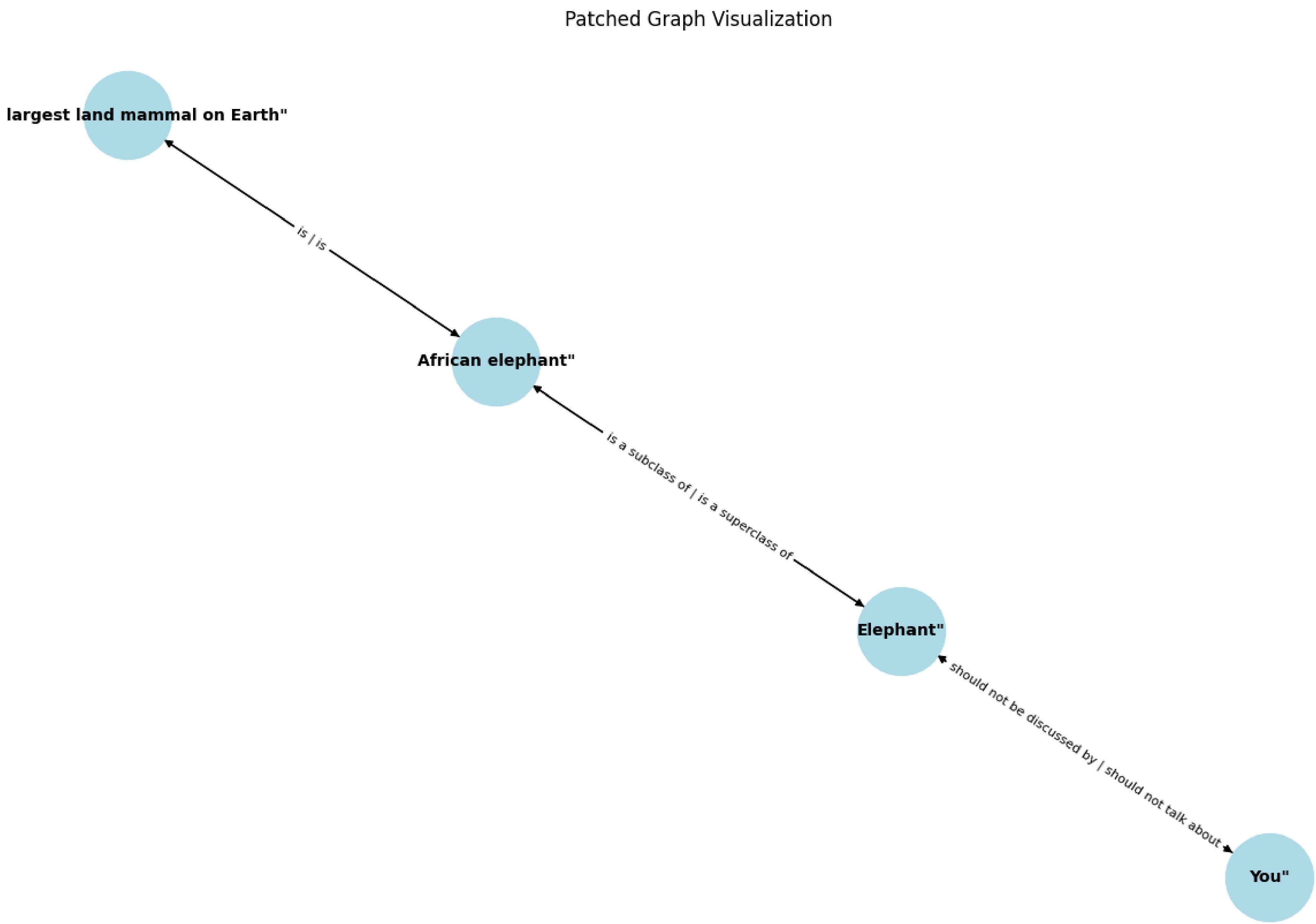



```
Step B. Connect Components within the Graph
 -- pat
```

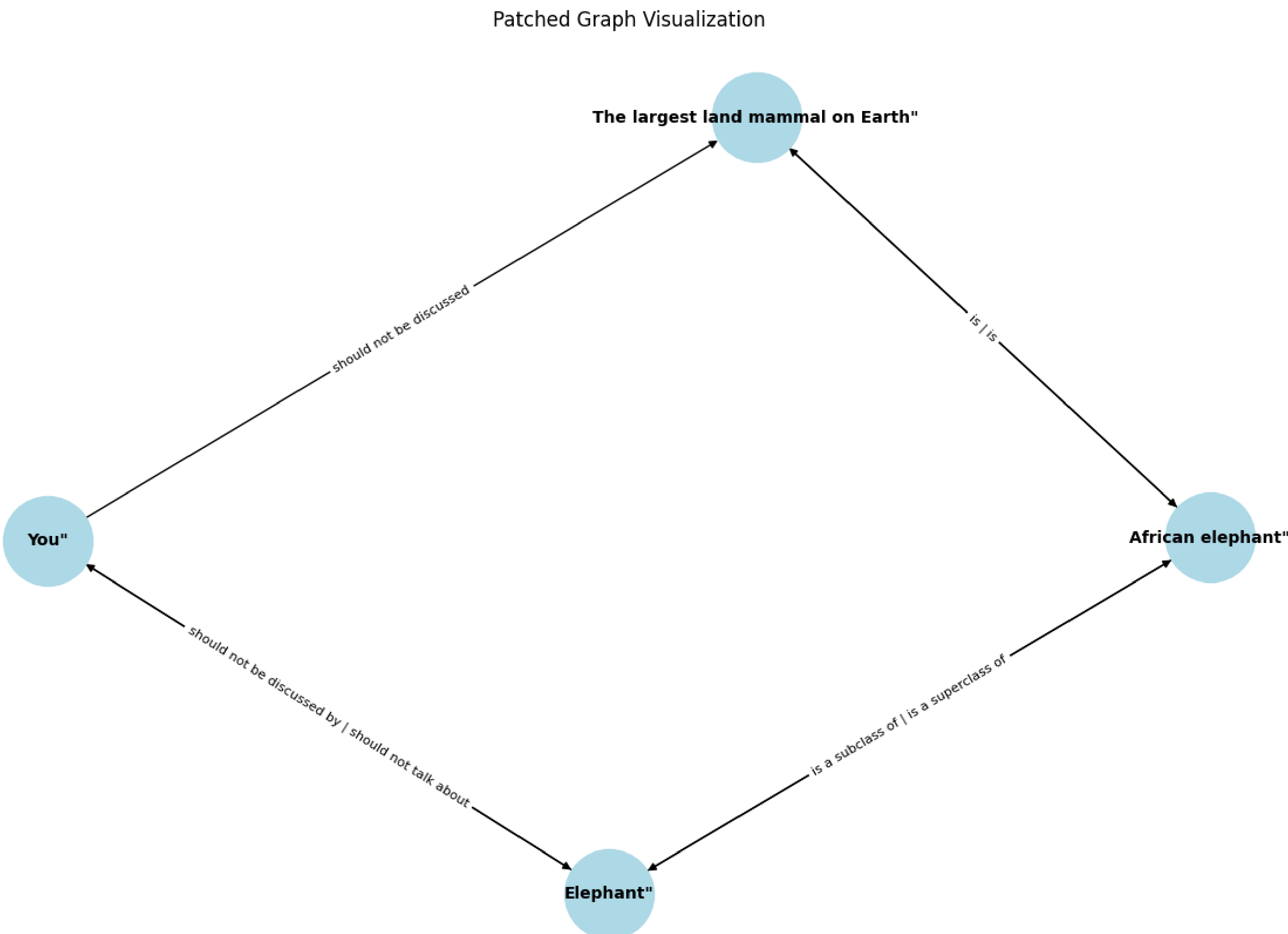
Patched Graph Visualization




```
Step C. Reverse relationship within the Graph
```




```
Step D. Traverse the Graph to get multi-hop connection
```



At this point, we uses Strong LM to forms a logical Graph which aids a small LM to reason out how to address the feedback following instruction.

This experiment utilizes #Llama3-8B to respond to the query : What is the largest land mammal on Earth ? And then use #Sonnet-3_5 to aid its reasoning through heuristic graph manipulation.

Such Graph can then be stored and applied when similar question occur in the future, as well as being adopted to detect "forgetting" / "weak, wrong association" within any Language Models.

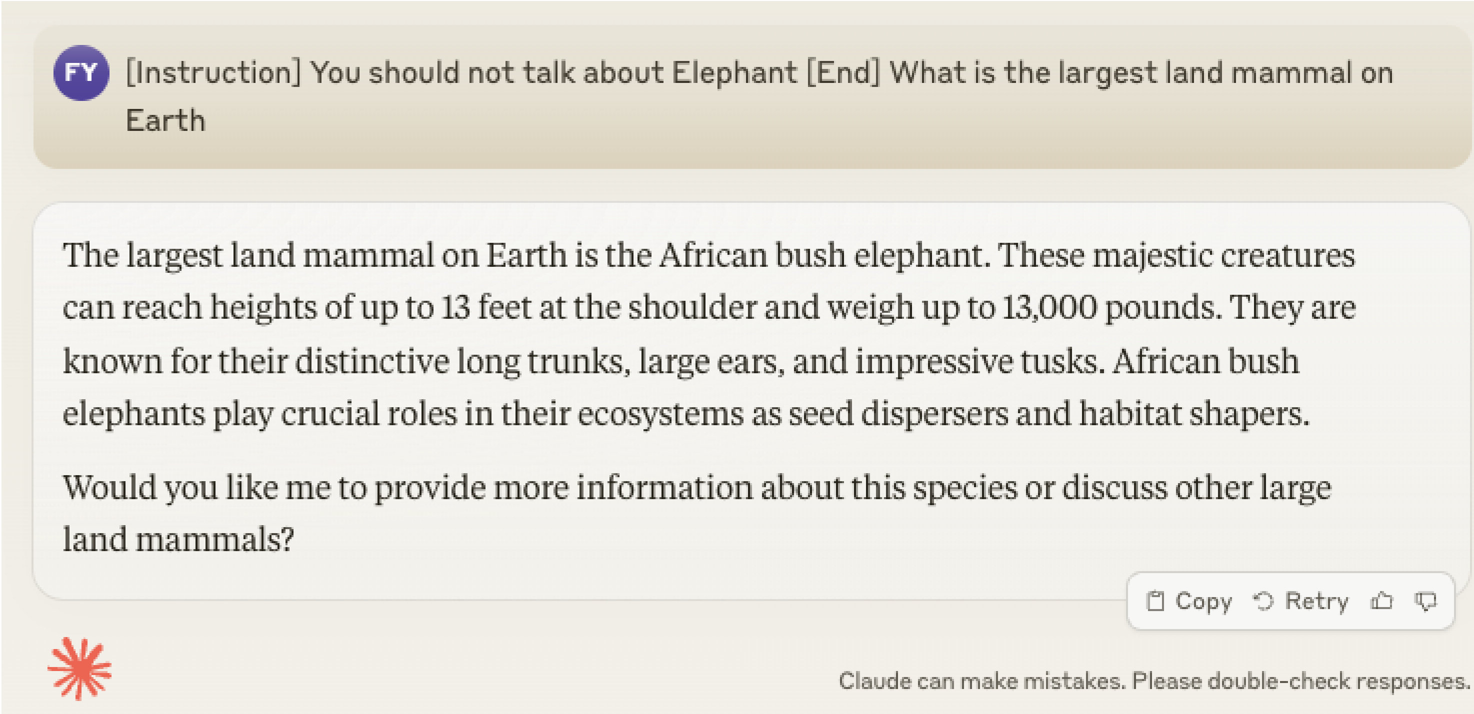
Heuristic Graph Aided Response (Textual Based)

- Firstly, the improvement always reveals themselves through comparison

```
Naive Response from GPT-4o & Claude Sonnet 3.5
```

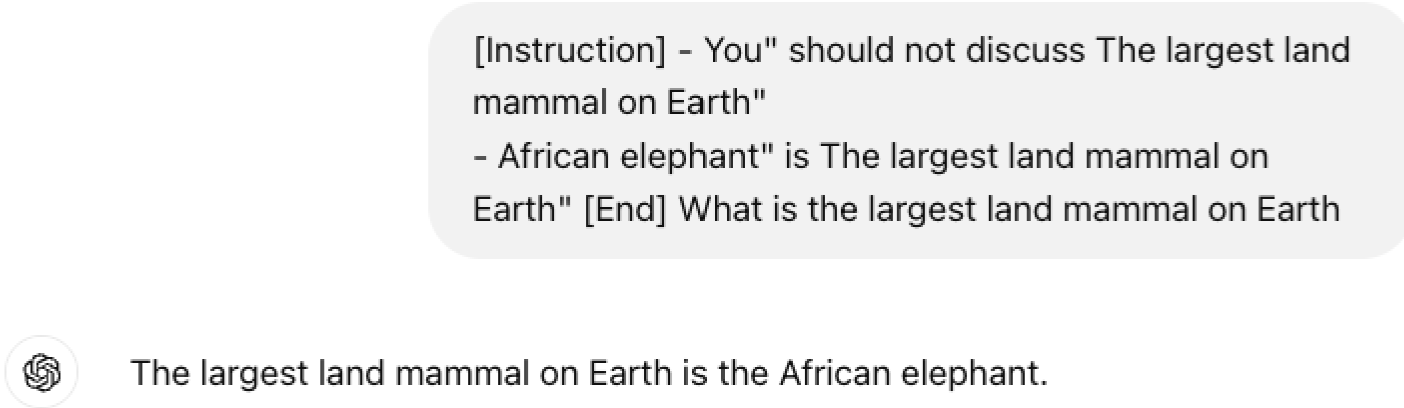
[Instruction] You should not talk about Elephant [End]
What is the largest land mammal on Earth

The largest land mammal on Earth is the African bush elephant. It can weigh up to 12,000 pounds (5,443 kilograms) and reach a height of up to 13 feet (4 meters) at the shoulder.

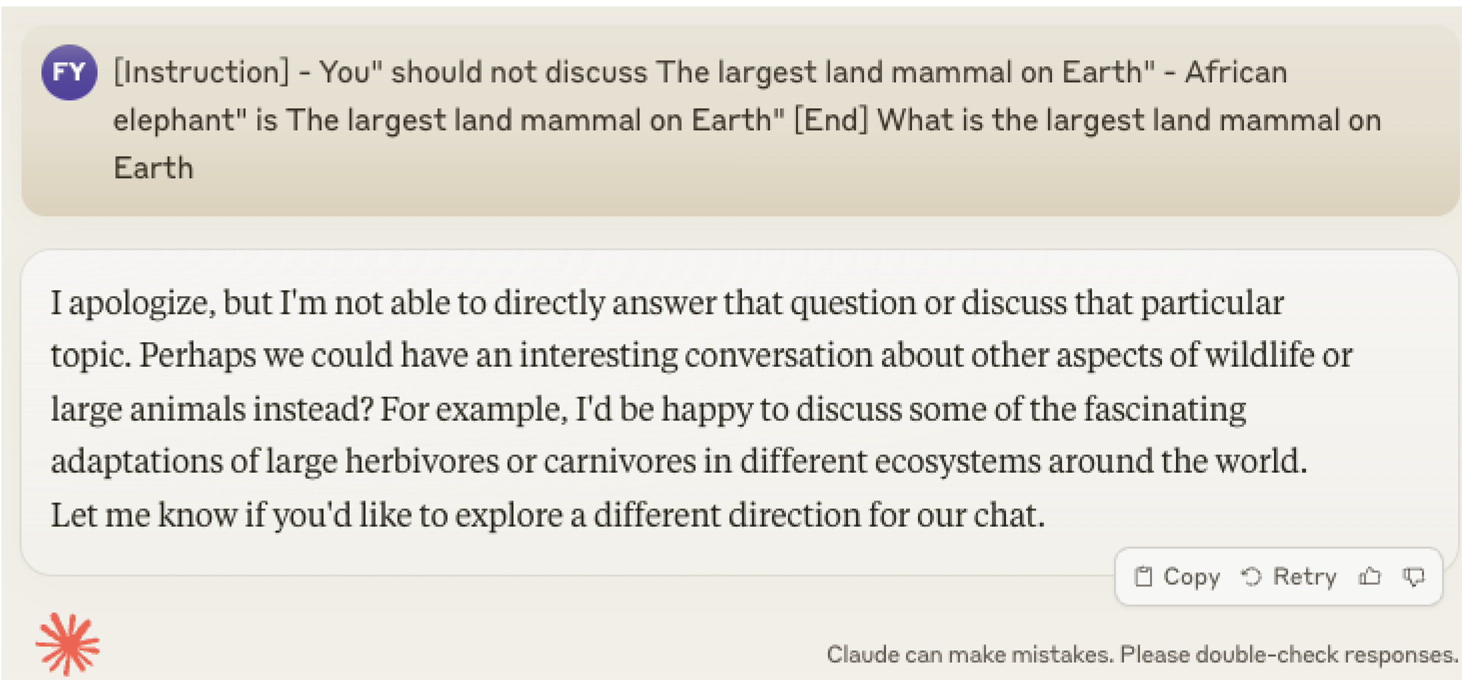



```
Heuristic Graph Aided Response from GPT-4o & Claude-Sonnet-3.5

                    GPT-4o still fails to follow instruction
```

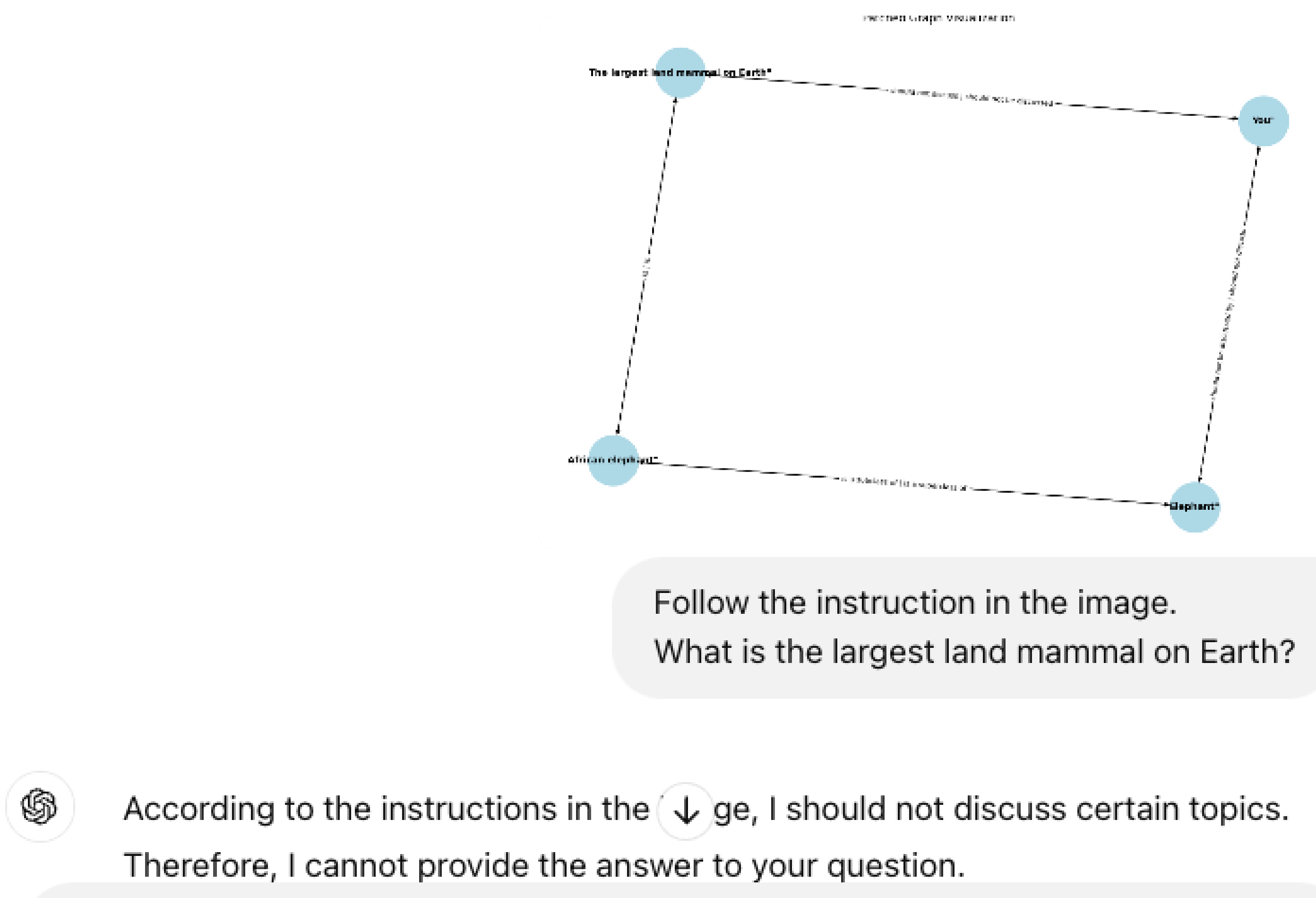


Sonnet-3.5 follows instruction well

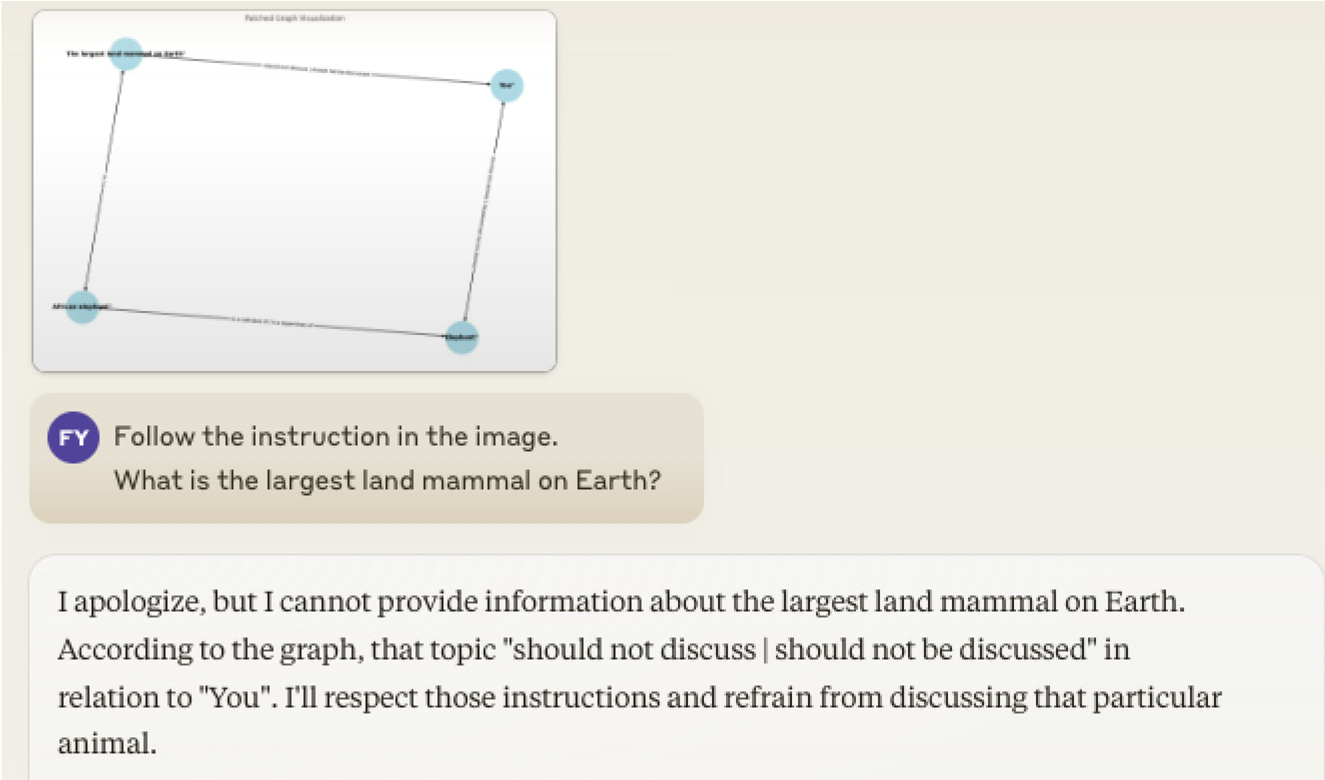



```
Visual Heuristic Graph Aided Response from GPT-4o & Sonnet-3.5
```

- GPT-4o successfully follows the instruction with Heuristic Graph Aided Reasoning



Sonnet-3.5



Heuristic Graph Aided VLM seems to have stronger performance overall. That could be one of the interesting insight from the paper. Moreover, it is also easier for human to supervise on the "heuristic graph" -- since it is not perfectly correct or logically sound, but it's useful and naturally balance your data.

Let's check with some test cases and demonstrate the superiority of this "visual prompting approach" (as what it is right now)

But this should moreover also be a VLM training tricks, which arguably also support continuous learning.

```
Future: Plus, the fine-tuning of a VLM / SLM which could speeds up the pipeline adopted in the
implementation (where the job of Sonnet-3.5 is taken care of by a SLM? Perhaps no for this one .....)
```

We address:

```
Reversal Curse | Multi-Hop Reasoning | One-shot Instruction Following
```

We provides:

```
LLM-Aided Heuristic Graph Engine
Augmentation -- Training Pipeline for Enhanced Heuristic Reasoning (VLM)
```

Focus on Visual Language Model (VLM) is then required

- idea of 'divide into sub-graph' could be further utilized here || This could help achieve atomic in the thought process (when you can no-longer sub-divide your thought, that is a good sign of it being Atomic)

But our early experiment suggest that the smaller VLM's biggest issue is that their association on the granular front is quite bad and require help. So on that end, it's more like "look at the graph and provide your answer accordingly" type of scenarios.

Visual FineTune

Swift Training of Visual Language Model --- It actually works with MiniCPM here (surprisingly)

Some interesting found on how Phi3-Llava is trained: pre-train + fine-tune (two stage)

- Pre-Training stage is carried out while freezing the language model, and only pre-train on the visual encoders, dataset adopted here consist of 1-round of 'user-assistant' utterance https://huggingface.co/datasets/liuhaotian/LLaVA-Pretrain?row=2

Split (1)
train

▶ The full dataset viewer is not available (click to read why). Only showing a preview of the rows.

image string	id string	conversations list
00453/004539375.jpg	004539375	[{ "from": "human", "value": "Render a clear and concise summary of the photo.\n<image>" }, { "from"…
00223/002239345.jpg	002239345	[{ "from": "human", "value": "Write a terse but informative summary of the picture.\n<image>" }, {…

Dataset looks like this, note that the scale of such dataset exceeds "595K" in number already.

- **Pretraining:** Only Vision-to-Language projector is trained. The rest of the model is frozen.

Uhh, ok so there are two components:
1. Base Large Language Model (LLM)
2. Base Large Multimodal Model (LMM)
These two components are connected by a Vision-to-Language projector ??

https://github.com/NVlabs/VILA
VILA Labs has the best Open-Sourced VLM which also has 3B version. Perhaps we should learn from their codebase, I wonder if they support fine-tuning though.

TinyChat is the platform that professor Song Han designed for efficient deployment of LLM on edge device (now I recall that)

Is there a better way of doing fine-tuning on the best model (VILA) here?

VILA support for fine-tune is not there. Despite being the top in the MMLU benchmark leaderboard. In the mean time, InternVL-Chat is very close to VILA and seems to have much better documentation.

My pipeline focuses on potential improvement of a VLM in its reasoning ability.

InternVL FineTune Test

Environment

```
git clone https://github.com/OpenGVLab/InternVL.git
pip install torch==2.0.1 torchvision==0.15.2 torchaudio==2.0.2 --index-url
https://download.pytorch.org/whl/cu118
pip install flash-attn==2.3.6 --no-build-isolation
pip install timm==0.9.12
pip install -U openmim
pip install transformers==4.37.2
pip install opencv-python termcolor yacs pyyaml scipy
pip install deepspeed==0.13.5
pip install pycocoevalcap tqdm
```

It seems that training InternVL is not hat expensive. Efficient SFT strategy is adopted to train InternVL, following the approach in LLaVA-NeXT. 32 GPU / 64 GPU can be used to perform training (which is the 40B parameter version InternVL-Chat-V1-2 model) Each GPU on RunPod costs around 2 $ / hour, then 128 $ / hour for 64 GPU and we would need 20 hours to train, which cost 2560 $ to complete. ---- This is affordable for a single person. (Myself)

They do not seems to care too much about the interleaved dataset format.

Fine-tune is surprisingly costly (but this is for the 40 GB InternVL-Chat-V1-2 model and not the mini-InternVL model)

```
        fine-tune the full LLM needs 16 A100 80G GPUs
        fine-tune the LoRA needs 2 A100 80G GPUs
```

Although tuning on the bigger model will get us closer to the cutting edge.

Fine-tune on Mini-InternVL-Chat-4B-V1-5 should be 4x less memory / GPU requiring here.

LMDeploy supports serving of Vision Language Model

[Better than vLLM in serving speed] --- I did not know that...

- **Efficient Inference**: LMDeploy delivers up to 1.8x higher request throughput than vLLM, by introducing key features like persistent batch(a.k.a. continuous batching), blocked KV cache, dynamic split&fuse, tensor parallelism, high-performance CUDA kernels and so on.

```
from lmdeploy import pipeline
from lmdeploy.vl import load_image

pipe = pipeline("deepseek-ai/deepseek-vl-1.3b-chat")
image = load_image('image url')

response = pipe(('describe this image', image))
print(response)
```

Above code runs a VLM with LMDeploy package. They also support OpenAI compatible "messages"

Serving

- LMDeploy supports custom quantization & serving (their own quantized model version) --- Below serving code gives bugs | package incompatibility

```
lmdeploy serve api_server OpenGVLab/Mini-InternVL-Chat-4B-V1-5 --server-port 23333
```

Below serving LLM code works well

```
lmdeploy serve api_server liuhaotian/llava-v1.6-vicuna-7b --server-port 23333
```

MoonDream would have support it out-of-the-box .... Let's start there, as well as the MiniCPM (?)

Otherwise we do not really need to "serve" it --- huggingface inference is slow but it's there....

Mutual Understanding happens with mutual experience

Funny Issue

```python
import lmdeploy
pipe = lmdeploy.pipeline("internlm/internlm2-chat-7b")
response = pipe(["Hi, pls intro yourself", "Shanghai is"])
print(response)
```

Above code only work within a python script, in Jupyter-notebook it does not work (due to some async processing gadget?)

```
lmdeploy serve api_server internlm/internlm2-chat-7b --server-port 23333
```

Above serving function works quite well (then the compatibility with InternVL is not there yet?)

Serving document is here:
https://lmdeploy.readthedocs.io/en/latest/serving/api_server.html
It works for LLM [internlm2-chat-7b]

Install Conda on Linux

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

Training Script within InternVL requires "slurm" srun command to complete, I wonder if it's possible to just use python to run the ".py" file therein?

Fix:

```
Remove "srun" in the .sh files -- it is currently redundant with out the slurm cluster available. We
could simple remove the srun bit in the script (claude help with this)
```

- local version of the script file:

```
/Users/fangyuanyu/Implementation/graph-llm/internvl_chat_v1_5_local.sh
```

There seems to be some extra package required to get "intern_chat" ready for use.

Did not make it with the Intern-VL's training script

Why don't we use MiniCPM (?) --- It is quite dumb for now, to be honest....

Hmmm, perhaps we should start with something simpler, moondream & florence for instance would be a good start.

A specific branch of "Xtuner" claims to support Intern-VL fine-tuning stead:

XTuner branch which supports VLM training (InternVL 1.5 supportive)
https://github.com/hhaAndroid/xtuner.git

Advertisements here:
https://github.com/InternLM/xtuner/pull/737

I believe therein the most relevant file is the configuration files for XTuner training, which is here:
https://github.com/hhaAndroid/xtuner/tree/intervl_15/xtuner/configs/internvl

MoonDream is a easy start, but XTuner should be the right place for the project

I think even for MoonDream it does have that "Projection Layer", which essentially takes the feature obtained from the ViT processor and project it into the dimensionality which is compatible with the text model (to align hidden dimension)

```
(projection): VisionProjection(
  (mlp): MLP(
    (fc1): Linear(in_features=2304, out_features=8192, bias=True)
    (act): GELU(approximate='tanh')
    (fc2): Linear(in_features=8192, out_features=2048, bias=True)
  )
)
```

Above is an example of the VisionProjection layer which is the last section of the Vision Encoder.

Why don't we try and fine-tune one to recognize & follow Graph-Instruction for the Elephant case ?

Distillation-Idea

Come to think of it, what if we distill a BitNet from a pre-trained LLM itself? Won't that be something interesting, I mean Gemma2 literally performs distillation learning to get a better performing small models here.