XrayGPT Training Code Tracing

Step 1: Entry Point & Imports

train.py

task = tasks.setup_task(cfg)

Calls : xraygpt/tasks/__init__.py :: setup_task()

Step 2 : Task Setup

xraygpt/tasks/__init__.py

task = registry.get_task_class(task_name).setup_task(cfg=cfg)

Calls : xraygpt/common/registry.py :: get_task_class()

Returns : ImageTextPretrainTask (registered in image_text_pretrain.py)

Step 3 : Task Registration

xraygpt/tasks/image_text_pretrain.py

@registry.register_task("image_text_pretrain")

Registers: ImageTextPretrainTask class in registry

Step 4 : Dataset Building

train.py

datasets = task.build_datasets(cfg)

Calls : xraygpt/tasks/base_task.py :: build_datasets()

xraygpt/tasks/base_task.py

builder = registry.get_builder_class(name)(dataset_config)

Calls : xraygpt/common/registry.py :: get_builder_class()

Returns : MIMICBuilder or OpenIBuilder

Step 5 : Dataset Builder Registration

xraygpt/datasets/builders/image_text_pair_builder.py

@registry.register_builder("mimic")

Registers : MIMICBuilder class

@registry.register_builder("openi")

Registers : OpenIBuilder class

Step 6 : Dataset Creation

xraygpt/datasets/builders/image_text_pair_builder.py

datasets['train'] = dataset_cls(...)

Calls : xraygpt/datasets/datasets/mimic_dataset.py :: MIMICDataset.__init__()

xraygpt/datasets/datasets/mimic_dataset.py

class MIMICDataset(CaptionDataset)

def __getitem__(self, index)

Called by: DataLoader during training

Step 7 : Model Building

train.py

model = task.build_model(cfg)

Calls : xraygpt/tasks/base_task.py :: build_model()

xraygpt/tasks/base_task.py

model_cls = registry.get_model_class(model_config.arch)

Calls : xraygpt/common/registry.py::get_model_class()

Returns: MiniGPT4 class

Step 8 : Model Registration

xraygpt/models/mini_gpt4.py

@registry.register_model("mini_gpt4")

Registers : MiniGPT4 class in registry

Step 9 : Runner Creation

train.py

runner_cls = registry.get_runner_class(cfg.run_cfg.get("runner","runner_base"))

Calls: **xraygpt/common/registry.py** :: get_runner_class()

Returns : RunnerBase class

runner = get_runner_class(cfg)(...)

Calls : xraygpt/runners/runner_base.py :: RunnerBase.__init__()

Step 10 : Runner Registration

train.py

runner.train()

Calls : xraygpt/runners/runner_base.py :: train()

xraygpt/runners/runner_base.py

train_stats = self.train_epoch(cur_epoch)

Calls : xraygpt/runners/runner_base.py :: train_epoch()

return self.task.train_epoch(...)

Calls: xraygpt/tasks/base_task.py :: train_epoch()

samples = next(data_loader)

Calls : xraygpt/datasets/datasets/mimic_dataset.py :: __getitem__()

Step 12 : Inner Training Loop

xraygpt/tasks/base_task.py

return self._train_inner_loop(...)

Calls : xraygpt/tasks/base_task.py :: _train_inner_loop()

image = self.vis_processor(image)

Calls : xraygpt/processors/blip_processors.py :: Blip2ImageTrainProcessor.__call__()

Calls : xraygpt/processors/blip_processors.py :: Blip2CaptionProcessor()

Step 13 : Data Preparation

xraygpt/tasks/base_task.py

samples = prepare_sample(samples, cuda_enabled=cuda_enabled)

Calls: xraygpt/datasets/data_utils.py : : prepare_sample()

samples = move_to_cuda(samples)

Calls : xraygpt/datasets/data_utils.py :: move_to_cuda()

Step 14 : Forward Pass

xraygpt/tasks/base_task.py

loss = self.train_step(model=model, samples=samples)

Calls : xraygpt/tasks/base_task.py :: train_step()

loss = model(samples)["loss"]

Calls : xraygpt/models/mini_gpt4.py :: forward()

xraygpt/models/mini_gpt4.py

img_embeds, atts_img = self.encode_img(image)

Calls: xraygpt/models/mini_gpt4.py :: encode_img()

prompt = random.choice(self.prompt_list)

Uses : prompts/alignment.txt (loaded in __init__)

img_embeds, atts_img = self.prompt_wrap(img_embeds, atts_img, prompt)

Calls : xraygpt/models/mini_gpt4.py :: prompt_wrap()

to_regress_tokens = self.llama_tokenizer(...)

Calls : LlamaTokenizer.from_pretrained() (HuggingFace)

outputs = self.llama_model(...)

Calls : LlamaForCausalLM.forward() (HuggingFace)