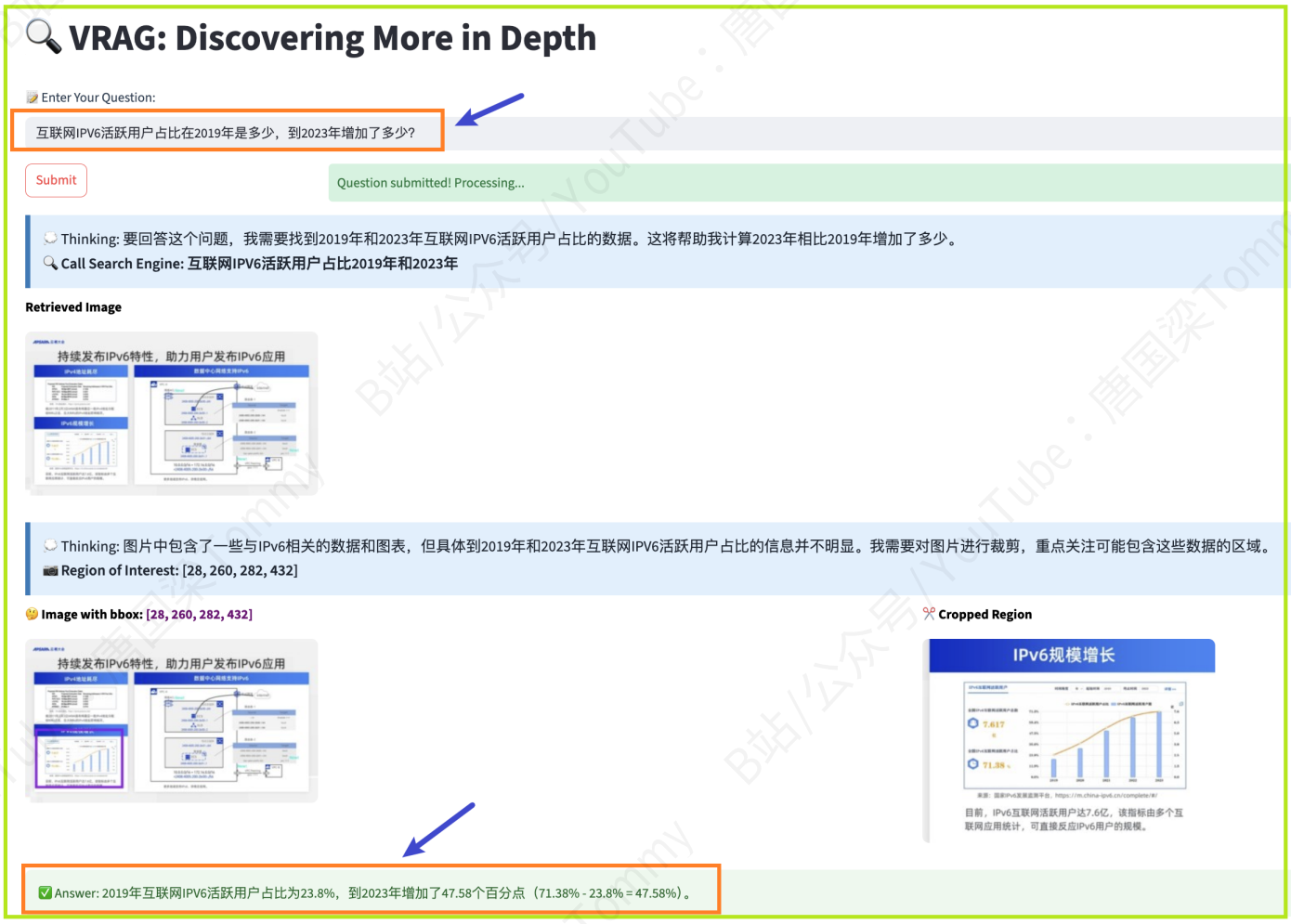
VRAG 项目全流程实战-快速入门

VRAG（Visual Retrieval-Augmented Generation）是一个面向多模态大模型（LLM+视觉）的检索增强生成（RAG）系统，通过粗粒度到细粒度的方法，聚焦于“视觉+文本”场景下的知识检索与智能问答。

核心功能:

1. **多轮对话**: 支持连续问答，保持上下文
2. **视觉搜索**: 根据查询检索相关图片
3. **图像裁剪**: 支持bbox坐标精确裁剪
4. **渐进推理**: 从粗粒度到细粒度的信息收集

使用示例:



1. 项目概览

1.1 核心组件

```
VRAG/
├── demo/                      # 演示系统
│   ├── app.py                 # Streamlit Web界面
│   └── vrag_agent.py          # VRAG智能体核心实现
├── search_engine/             # 视觉搜索引擎
│   ├── vl_embedding.py        # 多模态嵌入模型
│   ├── search_engine.py       # 搜索引擎核心
│   ├── search_engine_api.py # FastAPI服务
│   ├── ingestion.py           # 文档索引构建
│   └── corpus/                # 文档语料库
├── verl/                      # VRAG-RL训练框架
│   ├── trainer/               # 训练器配置
│   ├── models/                # 模型定义
│   └── workers/               # 分布式训练工作器
└── model_eval/                # 模型评估工具
```

1.2 技术栈

- **搜索引擎**: ColPali嵌入 (vidore/colqwen2-v1.0)
- **VLM模型**: Qwen2.5-VL-7B-VRAG / Qwen2.5-VL-7B-Instruct
- **训练框架**: VERL (基于Ray的分布式RL框架)
- **Web界面**: Streamlit + FastAPI
- **深度学习**: PyTorch + FSDP + Megatron

---

2. 实践指南

> **最低配置：**
>
> - GPU: 2张3090/4090 24GB 或同等性能显卡
> - 内存: 64GB RAM
> - 存储: 500GB 可用空间

2.1 环境配置与依赖安装

```
# 1. 创建conda环境
conda create -n vrag python=3.10
#如果是AutoDL，则需要执行下面的命令。
#conda init bash && source /root/.bashrc
conda activate vrag

# 2. 安装基础依赖
pip install torch==2.6.0

cd VRAG/

pip install -r requirements.txt
```

```
Successfully installed SQLAlchemy-2.0.41 accelerate-1.9.0 aiohappyeyeballs-2.6.1 aiohttp-3.12.14 aiohttp_cors-0.8.1 aiosignal-1.4.0 airportsdata-20250706 altair-5.5.0 annotated-types-0.7.0 antlr4-python3-runtime-4.9.3 anyio-4.9.0 astor-0.8.1 async-timeout-5.0.1 attrs-25.3.0 beautifulsoup4-4.13.4 blake3-1.0.5 blinker-1.9.0 cachetools-5.5.2 certifi-2025.7.14 charset_normalizer-3.4.2 click-8.2.1 cloudpickle-3.1.1 codetiming-1.4.0 colorful-0.5.7 colpali_engine-0.3.11.dev7+g481eb20 compressed-tensors-0.9.2 cupy-cuda12x-13.5.1 dataclasses-json-0.6.7 datasets-4.0.0 deprecated-1.2.18 depyf-0.18.0 dill-0.3.8 dirtyjson-1.0.8 diskcache-5.6.3 distlib-0.4.0 distro-1.9.0 dnspython-2.7.0 einops-0.8.1 email-validator-2.2.0 fastapi-0.116.1 fastapi-cli-0.0.8 fastapi-cloud-cli-0.1.4 fastrlock-0.8.3 filetype-1.2.0 frozenlist-1.7.0 fsspec-2025.3.0 gguf-0.10.0 gitdb-4.0.12 gitpython-3.1.45 google-api-core-2.25.1 google-auth-2.40.3 googleapis-common-protos-1.70.0 greenlet-3.2.3 grpcio-1.74.0 h11-0.16.0 hf-xet-1.1.5 httpcore-1.0.9 httptools-0.6.4 httpx-0.28.1 huggingface-hub-0.34.1 hydra-core-1.3.2 idna-3.10 importlib_metadata-8.7.0 interegular-0.3.3 jiter-0.10.0 joblib-1.5.1 jsonschema-4.25.0 jsonschema-specifications-2025.4.1 lark-1.2.2 latex2sympy2_extended-1.10.2 liger-kernel-0.6.0 llama-cloud-0.1.5 llama-index-0.12.0 llama-index-agent-openai-0.4.0 llama-index-cli-0.4.0 llama-index-core-0.12.1 llama-index-embeddings-huggingface-0.4.0 llama-index-embeddings-openai-0.3.0 llama-index-indices-managed-llama-cloud-0.6.2 llama-index-legacy-0.9.48.post4 llama-index-llms-openai-0.3.1 llama-index-multi-modal-llms-openai-0.3.0 llama-index-program-openai-0.3.0 llama-index-question-gen-openai-0.3.0 llama-index-readers-file-0.4.0 llama-index-readers-llama-parse-0.4.0 llama-parse-0.5.14 llguidance-0.7.30 llvmlite-0.43.0 lm-format-enforcer-0.10.11 markdown-it-py-3.0.0 marshmallow-3.26.1 math-verify-0.8.0 mdurl-0.1.2 mistral_common-1.8.3 msgpack-1.1.1 msgspec-0.19.0 multidict-6.6.3 multiprocess-0.70.16 mypy-extensions-1.1.0 narwhals-1.48.1 ninja-1.11.1.4 nltk-3.9.1 numba-0.60.0 numpy-1.26.4 omegaconf-2.3.0 openai-1.97.1 opencensus-0.11.4 opencensus-context-0.1.3 opencv-python-headless-4.11.0.86 opentelemetry-api-1.35.0 opentelemetry-exporter-prometheus-0.56b0 opentelemetry-proto-1.35.0 opentelemetry-sdk-1.35.0 opentelemetry-semantic-conventions-0.56b0 orjson-3.11.1 outlines-0.1.11 outlines_core-0.1.26 pandas-2.3.1 partial-json-parser-0.2.1.1.post6 peft-0.15.2 pillow-11.3.0 prometheus-fastapi-instrumentator-7.1.0 prometheus_client-0.22.1 propcache-0.3.2 proto-plus-1.26.1 protobuf-6.31.1 py-cpuinfo-9.0.0 py-spy-0.4.0 pyarrow-21.0.0 pyasn1-0.6.1 pyasn1-modules-0.4.2 pybind11-3.0.0 pycountry-24.6.1 pydantic-2.9.2 pydantic-core-2.23.4 pydantic-extra-types-2.10.5 pydeck-0.9.1 pylatexenc-2.10 pypdf-5.8.0 python-dotenv-1.1.1 python-json-logger-3.3.0 python-multipart-0.0.20 pytz-2025.2 pyyaml-6.0.2 ray-2.48.0 referencing-0.36.2 regex-2024.11.6 requests-2.32.4 rich-14.1.0 rich-toolkit-0.14.8 rignore-0.6.4 rpds-py-0.26.0 rsa-4.9.1 safetensors-0.5.3 scikit-learn-1.7.1 scipy-1.15.3 sentence-transformers-5.0.0 sentencepiece-0.2.0 sentry-sdk-2.33.2 shellingham-1.5.4 smart_open-7.3.0.post1 smmap-5.0.2 sniffio-1.3.1 soupsieve-2.7 starlette-0.47.2 streamlit-1.47.1 striprtf-0.0.26 tenacity-8.5.0 tensordict-0.5.0 threadpoolctl-3.6.0 tiktoken-0.9.0 tokenizers-0.21.2 toml-0.10.2 torchaudio-2.6.0 torchdata-0.11.0 torchvision-0.21.0 tqdm-4.67.1 transformers-4.51.3 typer-0.16.0 typing-inspect-0.9.0 tzdata-2025.2 urllib3-2.5.0 uvicorn-0.35.0 uvloop-0.21.0 virtualenv-20.32.0 vllm-0.8.2 wandb-0.21.0 watchdog-6.0.0 watchfiles-1.1.0 websockets-15.0.1 wrapt-1.17.2 xformers-0.0.29.post2 xgrammar-0.1.16 xxhash-3.5.0 yarl-1.20.1 zipp-3.23.0
```

```
# 3. 下载 Qwen2.5-VL-7B-VRAG

pip install huggingface_hub

# 需要登录
huggingface-cli login

# 设置国内镜像
# export HF_ENDPOINT=https://hf-mirror.com

hf download autumncc/Qwen2.5-VL-7B-VRAG --local-dir /root/models/Qwen2.5-VL-7B-VRAG
hf download Qwen/Qwen2.5-VL-3B-Instruct --local-dir /root/models/Qwen2.5-VL-3B-Instruct
```

```
hf download Qwen/Qwen2.5-1.5B-Instruct --local-dir /root/models/Qwen2.5-1.5B-Instruct
hf download vidore/colqwen2-v1.0 --local-dir /root/models/colqwen2-v1.0
```

2.2 快速上手

> 使用预训练模型的演示

2.2.1 启动检索引擎服务

```
python search_engine/search_engine_api.py
# 服务启动后监听：http://0.0.0.0:8002/search

# ⚠️注意：
# 自动下载的（vidore/colqwen2-v1.0）模型通常存储在以下路径下：~/.cache/huggingface/hub/
```

搜索引擎核心原理:

- 使用ColQwen2模型进行视觉-文本嵌入
- 预加载所有图片的嵌入向量到GPU内存
- 支持批量查询，返回Top-K相似图片

2.2.2 启动 VLM 推理服务

```
CUDA_VISIBLE_DEVICES=0 vllm serve /root/models/Qwen2.5-VL-7B-VRAG --port 8001 --host
0.0.0.0 --limit-mm-per-prompt image=5 --served-model-name Qwen/Qwen2.5-VL-7B-Instruct --
max-model-len 8192 --cpu-offload-gb 4

# 服务启动后监听：http://0.0.0.0:8002/search
```

模型选择说明:

- `autumncc/Qwen2.5-VL-7B-VRAG`: 官方微调的VRAG专用模型
- `Qwen/Qwen2.5-VL-7B-Instruct`: Qwen官方基础多模态模型

2.2.3 启动 Web 界面

```
# 启动Streamlit演示界面
streamlit run demo/app.py --server.port 8800 --server.address 0.0.0.0
```

- **一键启动所有服务**

```
bash run_demo.sh
```

这将启动三个服务：

- 搜索引擎API（端口8002）
- VLLM服务（端口8001）- 运行Qwen2.5-VL-7B-VRAG模型
- Streamlit演示应用（默认端口8501）
