Contents

1 Rasa 智能训练平台部署指南 **1**

1.1 目录 . . . 1
1.2 系统概述 . . . 1
1.2.1 技术栈 . . . 1
1.3 硬件要求 . . . 2
1.3.1 最低配置 (开发环境) . . . 2
1.3.2 推荐配置 (生产环境) . . . 2
1.3.3 高性能配置 (大规模部署) . . . 2
1.4 环境准备 . . . 2
1.4.1 操作系统要求 . . . 2
1.4.2 基础软件安装 . . . 2
1.5 GPU 配置 . . . 3
1.5.1 NVIDIA GPU 环境 (推荐 RTX 3080Ti 及以上) . . . 3
1.5.2 RTX 3080Ti 专用优化 . . . 3
1.6 部署步骤 . . . 4
1.6.1 1. 克隆项目 . . . 4
1.6.2 2. 后端部署 . . . 4
1.6.3 3. 前端部署 . . . 4
1.6.4 4. Rasa 服务部署 . . . 4
1.6.5 5. Nginx 反向代理配置 . . . 5
1.7 性能优化 . . . 5
1.7.1 1. Rasa 模型优化 . . . 5
1.7.2 2. 系统级优化 . . . 5
1.7.3 3. Python 优化 . . . 6
1.7.4 4. 数据库优化 . . . 6
1.8 监控与维护 . . . 6
1.8.1 1. 系统监控 . . . 6
1.8.2 2. 日志管理 . . . 6
1.8.3 3. 健康检查 . . . 6
1.9 故障排除 . . . 7
1.9.1 常见问题 . . . 7
1.9.2 性能基准 . . . 7
1.10 扩展部署 . . . 8
1.10.1 Docker 容器化 . . . 8
1.10.2 Kubernetes 部署 . . . 8

1 Rasa 智能训练平台部署指南

1.1 目录

- 系统概述
- 硬件要求
- 环境准备
- GPU 配置
- 部署步骤
- 性能优化
- 监控与维护
- 故障排除

1.2 系统概述

Rasa 智能训练平台是一个基于 FastAPI + React + Rasa 的 NLP 训练和推理系统，支持意图识别、实体提取、模型训练等功能。

1.2.1 技术栈

- **后端**: FastAPI + SQLAlchemy + SQLite

- **前端**: React + Antd
- **NLP**: Rasa 3.x + TensorFlow
- **数据库**: SQLite (可升级为 PostgreSQL)

1.3 硬件要求

1.3.1 最低配置（开发环境）

- **CPU**: 4 核心 Intel i5 或 AMD Ryzen 5
- **内存**: 8GB RAM
- **存储**: 20GB 可用空间
- **GPU**: 可选 (CPU 模式)

1.3.2 推荐配置（生产环境）

- **CPU**: 8 核心 Intel i7/Xeon 或 AMD Ryzen 7/Threadripper
- **内存**: 16GB RAM
- **存储**: 50GB SSD
- **GPU**: NVIDIA GTX 1060 6GB 或更高 (支持 CUDA 11.0+)

1.3.3 高性能配置（大规模部署）

- **CPU**: 16 核心 Intel Xeon 或 AMD EPYC
- **内存**: 32GB+ RAM
- **存储**: 100GB+ NVMe SSD
- **GPU**: NVIDIA RTX 3080/4080 或 Tesla T4/V100
- **网络**: 千兆以太网

1.4 环境准备

1.4.1 操作系统要求

- **Linux**: Ubuntu 20.04+ / CentOS 8+ (推荐)
- **Windows**: Windows 10/11 + WSL2 (开发)
- **macOS**: macOS 11+ (开发)

1.4.2 基础软件安装

```
# 更新系统
sudo apt update && sudo apt upgrade -y

# 安装 Python 3.8+
sudo apt install python3.8 python3.8-dev python3-pip -y

# 安装 Node.js 16+
curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash -
sudo apt install nodejs -y

# 安装 Git 和其他工具
sudo apt install git build-essential curl wget -y
```

1.4.2.1 Linux (Ubuntu/Debian)

```
# 安装 Chocolatey 包管理器
Set-ExecutionPolicy Bypass -Scope Process -Force
iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

# 安装 Python 和 Node.js
choco install python nodejs git -y
```

1.4.2.2 Windows

1.5 GPU 配置

1.5.1 NVIDIA GPU 环境 (推荐 RTX 3080Ti 及以上)

```
# Ubuntu 自动安装推荐驱动
sudo ubuntu-drivers autoinstall

# 或手动安装最新驱动
sudo apt install nvidia-driver-515 -y
```

1.5.1.1 1. 安装 NVIDIA 驱动

```
# 下载 CUDA 11.8 (Rasa 兼容版本)
wget https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda_11.8.0_520.61.05_linux.run
sudo sh cuda_11.8.0_520.61.05_linux.run

# 配置环境变量
echo 'export PATH=/usr/local/cuda-11.8/bin:$PATH' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=/usr/local/cuda-11.8/lib64:$LD_LIBRARY_PATH' >> ~/.bashrc
source ~/.bashrc
```

1.5.1.2 2. 安装 CUDA Toolkit

```
# 下载 cuDNN 8.6+ for CUDA 11.x
# 从 NVIDIA 官网下载后解压
sudo cp cudnn-*/include/* /usr/local/cuda/include/
sudo cp cudnn-*/lib64/* /usr/local/cuda/lib64/
sudo chmod a+r /usr/local/cuda/include/cudnn*.h /usr/local/cuda/lib64/libcudnn*
```

1.5.1.3 3. 安装 cuDNN

```
# 检查 NVIDIA 驱动
nvidia-smi

# 检查 CUDA
nvcc --version

# 检查 TensorFlow GPU 支持
python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"
```

1.5.1.4 4. 验证 GPU 环境

1.5.2 RTX 3080Ti 专用优化

```
# 设置 GPU 内存增长模式
export TF_FORCE_GPU_ALLOW_GROWTH=true

# 设置 CUDA 可见设备
export CUDA_VISIBLE_DEVICES=0

# 优化 GPU 内存分配
export TF_GPU_ALLOCATOR=cuda_malloc_async
```

1.6 部署步骤

1.6.1 1. 克隆项目

```
git clone <project-repository>
cd instruction_training_platform
```

1.6.2 2. 后端部署

```
cd backend

# 创建虚拟环境
python3 -m venv venv
source venv/bin/activate  # Linux/Mac
# venv\Scripts\activate   # Windows

# 安装依赖
pip install -r requirements.txt

# GPU 版本 TensorFlow (如果有 GPU)
pip uninstall tensorflow
pip install tensorflow-gpu==2.11.0

# 初始化数据库
python -c "from database import engine, Base; Base.metadata.create_all(bind=engine)"

# 启动后端服务
uvicorn app:app --host 0.0.0.0 --port 8000 --workers 4
```

1.6.3 3. 前端部署

```
cd frontend

# 安装依赖
npm install

# 生产构建
npm run build

# 使用 Nginx 部署 (推荐)
sudo cp -r build/* /var/www/html/

# 或使用 Node.js 服务
npm install -g serve
serve -s build -l 3000
```

1.6.4 4. Rasa 服务部署

```
cd rasa

# 安装 Rasa (GPU 版本)
pip install rasa[spacy]==3.6.0
pip install tensorflow-gpu==2.11.0

# 训练模型
rasa train --config config.yml --domain data/domain.yml --data data/

# 启动 Rasa 服务器 (GPU 模式)
```

```
CUDA_VISIBLE_DEVICES=0 rasa run --enable-api --cors "*" --port 5005 --debug
```

1.6.5 5. Nginx 反向代理配置

```
# /etc/nginx/sites-available/rasa-platform
server {
    listen 80;
    server_name your-domain.com;

    # 前端静态文件
    location / {
        root /var/www/html;
        try_files $uri $uri/ /index.html;
    }

    # 后端API
    location /api/ {
        proxy_pass http://localhost:8000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Rasa API
    location /rasa/ {
        proxy_pass http://localhost:5005/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

1.7 性能优化

1.7.1 1. Rasa 模型优化

- **模型大小**: 使用较小的 transformer_size (64-128)
- **层数**: 使用 1-2 层 transformer
- **批次大小**: GPU 环境下使用 128+
- **特征限制**: 限制 max_features 减少内存使用

1.7.2 2. 系统级优化

```
# 增加文件描述符限制
echo "* soft nofile 65536" >> /etc/security/limits.conf
echo "* hard nofile 65536" >> /etc/security/limits.conf

# 优化网络参数
echo "net.core.somaxconn = 65536" >> /etc/sysctl.conf
echo "net.ipv4.tcp_max_syn_backlog = 65536" >> /etc/sysctl.conf
sysctl -p
```

1.7.3 3. Python 优化

```
# 使用更快的 JSON 库
pip install ujson

# 启用 Python 优化
export PYTHONOPTIMIZE=2

# 使用更快的异步库
pip install uvloop
```

1.7.4 4. 数据库优化

```
-- 为数据库添加索引
CREATE INDEX idx_intent_name ON intents(intent_name);
CREATE INDEX idx_utterance_intent ON utterances(intent_id);
CREATE INDEX idx_response_intent ON responses(intent_id);
```

1.8 监控与维护

1.8.1 1. 系统监控

```
# 安装监控工具
sudo apt install htop iotop nvidia-ml-py3

# GPU 监控脚本
#!/bin/bash
while true; do
    nvidia-smi --query-gpu=timestamp,temperature.gpu,utilization.gpu,memory.used,memory.total --format=csv
    sleep 5
done
```

1.8.2 2. 日志管理

```
# 配置轮转日志
import logging.handlers

handler = logging.handlers.RotatingFileHandler(
    'rasa_platform.log',
    maxBytes=10*1024*1024,  # 10MB
    backupCount=5
)
```

1.8.3 3. 健康检查

```
# 创建健康检查脚本
#!/bin/bash
# health_check.sh

# 检查后端服务
curl -f http://localhost:8000/health || exit 1

# 检查 Rasa 服务
curl -f http://localhost:5005/status || exit 1

# 检查 GPU 状态
nvidia-smi || echo "GPU 检查失败"
```

1.9 故障排除

1.9.1 常见问题

```
# 检查驱动安装
lsmod | grep nvidia

# 重装 NVIDIA 驱动
sudo apt purge nvidia-*
sudo apt autoremove
sudo ubuntu-drivers autoinstall
sudo reboot
```

1.9.1.1 1. GPU 未被识别

```
# 减少模型复杂度
# 在 config.yml 中调整参数
transformer_size: 32
number_of_transformer_layers: 1
epochs: 20

# 检查内存使用
free -h
nvidia-smi
```

1.9.1.2 2. Rasa 训练慢/失败

1.9.1.3 3. 推理速度慢

- 确认 GPU 正在使用: `nvidia-smi`
- 减少模型大小和复杂度
- 使用更快的相似度计算方法
- 启用模型缓存

```
# 增加虚拟内存
sudo fallocate -l 4G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile

# 减少模型批次大小
batch_size: 32  # 从 128 减少到 32
```

1.9.1.4 4. 内存不足

1.9.2 性能基准

1.9.2.1 目标性能指标 (RTX 3080Ti)

- **意图预测**: < 100ms (95% 的请求)
- **模型训练**: < 10 分钟 (1000 条数据)
- **并发处理**: 100+ QPS
- **内存使用**: < 4GB
- **GPU 利用率**: $> 80\%$ (训练时)

```
# 实时性能监控
watch -n 1 'nvidia-smi | grep -A 3 "GPU-Util"'
watch -n 1 'curl -s http://localhost:8000/api/health | jq .'
```

1.9.2.2 监控命令

1.10 扩展部署

1.10.1 Docker 容器化

```
# Dockerfile.rasa
FROM nvidia/cuda:11.8-runtime-ubuntu20.04

RUN apt-get update && apt-get install -y python3 python3-pip
COPY requirements.txt .
RUN pip3 install -r requirements.txt

COPY . /app
WORKDIR /app

CMD ["rasa", "run", "--enable-api", "--cors", "*", "--port", "5005"]
```

1.10.2 Kubernetes 部署

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rasa-platform
spec:
  replicas: 3
  selector:
    matchLabels:
      app: rasa-platform
  template:
    metadata:
      labels:
        app: rasa-platform
    spec:
      containers:
      - name: rasa
        image: rasa-platform:latest
        resources:
          limits:
            nvidia.com/gpu: 1
            memory: 4Gi
          requests:
            memory: 2Gi
```

注意：本文档会随系统更新而更新，建议定期查看最新版本。