使用 uv 管理 Python 环境指南

本指南介绍如何使用 `uv` 来管理本项目的 Python 环境。本项目已经配置了 `pyproject.toml` 文件，可以直接使用。

1. 安装 uv

Windows (PowerShell)

```
powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
```

macOS / Linux

```
curl -lsSf https://astral.sh/uv/install.sh | sh
```

安装完成后，请重启终端。

2. 初始化环境与安装依赖

在项目根目录下（即包含 `pyproject.toml` 的目录），运行以下命令会自动创建虚拟环境并按照锁文件（如果不存在则会生成 `uv.lock`）安装所有依赖：

```
uv sync
```

该命令会创建一个 `.venv` 目录，其中包含隔离的 Python 环境。

3. 常用命令

激活虚拟环境

虽然 `uv run` 可以自动使用虚拟环境，但如果你想在终端中激活它：

- **Windows (PowerShell):** `.venv\Scripts\activate`
- **macOS/Linux:** `source .venv/bin/activate`

添加新依赖

如果需要添加新的库（例如 `numpy`）：

```
uv add numpy
```

这会自动更新 `pyproject.toml` 和 `uv.lock`。

移除依赖

```
uv remove numpy
```

更新依赖

```
uv lock --upgrade
```

运行脚本

使用 `uv run` 可以在不需要手动激活虚拟环境的情况下运行脚本，它会自动使用项目环境：

```
uv run src/main.py
```

（假设你的入口文件是 `src/main.py`）

或者运行工具命令：

```
uv run gradio app.py
```

4. 依赖列表说明

当前 `pyproject.toml` 中包含的核心库：

- **深度学习**: `torch`, `transformers`, `huggingface-hub`
- **数据处理**: `pandas`
- **Web 前端**: `gradio`
- **大模型 API**: `openai` (预留), `python-dotenv`

5. 项目结构建议

建议将源码放在 `src/` 目录下，并确保 `pyproject.toml` 位于项目根目录。

```
项目根目录/
├── pyproject.toml  # 项目配置文件
├── uv.lock         # 依赖版本锁定文件（运行 uv sync 后生成）
├── .venv/          # 虚拟环境目录（自动生成）
├── data/           # 数据目录
└── src/            # 源代码
    └── main.py
```