Setup Instructions for Resume Tailor Agent

This document provides detailed setup instructions for different environments and use cases.

🖥️ System Requirements

Minimum Requirements

- **OS**: Windows 10+, macOS 10.14+, or Linux (Ubuntu 18.04+)
- **Python**: 3.8 or higher
- **RAM**: 4GB (8GB recommended for local LLM)
- **Storage**: 2GB free space (more for local LLM models)
- **Internet**: Required for API-based LLMs

Recommended Requirements

- **OS**: macOS 12+ or Ubuntu 20.04+
- **Python**: 3.10 or higher
- **RAM**: 16GB (for optimal local LLM performance)
- **Storage**: 10GB free space
- **CPU**: Multi-core processor (for local LLM)

🐍 Python Environment Setup

Option 1: Using venv (Recommended)

Bash

```bash
# Create virtual environment
python3 -m venv resume-tailor-env

# Activate environment
# On macOS/Linux:
source resume-tailor-env/bin/activate
# On Windows:
resume-tailor-env\Scripts\activate
```

```bash
# Install dependencies
pip install -r requirements.txt
```

Option 2: Using conda

Bash

```bash
# Create conda environment
conda create -n resume-tailor python=3.10

# Activate environment
conda activate resume-tailor

# Install dependencies
pip install -r requirements.txt
```

Option 3: Using pipenv

Bash

```bash
# Install pipenv if not already installed
pip install pipenv

# Install dependencies and create environment
pipenv install -r requirements.txt

# Activate environment
pipenv shell
```

🤖 LLM Setup Options

Option 1: Local Ollama (Free, Private)

macOS Installation

Bash

```bash
# Install Ollama
brew install ollama

# Or download from https://ollama.ai/download

# Start Ollama service
```

```
ollama serve

# Pull a model (in another terminal)
ollama pull mistral
```

Linux Installation

Bash

```bash
# Install Ollama
curl -fsSL https://ollama.ai/install.sh | sh

# Start Ollama service
ollama serve

# Pull a model (in another terminal)
ollama pull mistral
```

Windows Installation

1. Download Ollama from https://ollama.ai/download
2. Run the installer
3. Open Command Prompt or PowerShell
4. Run: `ollama serve`
5. In another terminal: `ollama pull mistral`

Recommended Models

Bash

```bash
# Lightweight, fast
ollama pull mistral

# More capable, larger
ollama pull llama2

# Code-focused
ollama pull codellama

# Conversational
ollama pull neural-chat
```

Option 2: OpenAI API (Paid, High Quality)

Setup Steps

1. Create account at https://platform.openai.com/
2. Generate API key in your dashboard
3. Set environment variable:

macOS/Linux:

Bash

```bash
export OPENAI_API_KEY="your_api_key_here"
# Add to ~/.bashrc or ~/.zshrc for persistence
echo 'export OPENAI_API_KEY="your_api_key_here"' >> ~/.bashrc
```

Windows:

Plain Text

```
set OPENAI_API_KEY=your_api_key_here
# Or use System Properties > Environment Variables for persistence
```

Available Models

- `gpt-3.5-turbo` : Fast, cost-effective
- `gpt-4` : Higher quality, more expensive
- `gpt-4-turbo-preview` : Latest capabilities

Option 3: Anthropic Claude (Paid, Excellent Reasoning)

Setup Steps

1. Create account at https://console.anthropic.com/
2. Generate API key
3. Set environment variable:

macOS/Linux:

Bash

```bash
export ANTHROPIC_API_KEY="your_api_key_here"
```

```
echo 'export ANTHROPIC_API_KEY="your_api_key_here"' >> ~/.bashrc
```

Windows:

```
Plain Text

set ANTHROPIC_API_KEY=your_api_key_here
```

Available Models

- `claude-3-haiku-20240307` : Fast, economical
- `claude-3-sonnet-20240229` : Balanced performance
- `claude-3-opus-20240229` : Highest capability

🌐 Web Interface Setup

Basic Setup

```bash
# Start Streamlit app
streamlit run streamlit_app.py

# Access at http://localhost:8501
```

Custom Configuration

```bash
# Custom port and host
streamlit run streamlit_app.py --server.port 8080 --server.address 0.0.0.0

# Disable usage stats collection
streamlit run streamlit_app.py --browser.gatherUsageStats false
```

Production Deployment

For production deployment, consider:

- Using a reverse proxy (nginx)
- Setting up SSL certificates

- Configuring authentication
- Using a process manager (PM2, systemd)

🔧 Configuration Files

Environment Variables (.env file)

Create a `.env` file in the project root:

Plain Text

```
# LLM API Keys
OPENAI_API_KEY=your_openai_key_here
ANTHROPIC_API_KEY=your_anthropic_key_here

# Ollama Configuration
OLLAMA_HOST=http://localhost:11434
OLLAMA_MODEL=mistral

# Application Settings
DEFAULT_MODEL=local
LOG_LEVEL=INFO
```

Streamlit Configuration

Create `.streamlit/config.toml`:

Plain Text

```
[server]
port = 8501
address = "0.0.0.0"
maxUploadSize = 200

[browser]
gatherUsageStats = false

[theme]
primaryColor = "#1f77b4"
backgroundColor = "#ffffff"
secondaryBackgroundColor = "#f0f2f6"
textColor = "#262730"
```

🧪 Testing Your Setup

1. Test Python Environment

Bash

```bash
python --version  # Should be 3.8+
pip list | grep -E "(streamlit|openai|anthropic|python-docx)"
```

2. Test Resume Parsing

Bash

```bash
python test_parser.py
```

Expected output: All tests should pass

3. Test LLM Connections

Bash

```bash
# Test local Ollama
python src/main.py --test-llm --model local

# Test OpenAI (if configured)
python src/main.py --test-llm --model openai

# Test Anthropic (if configured)
python src/main.py --test-llm --model anthropic
```

4. Test Web Interface

Bash

```bash
streamlit run streamlit_app.py
```

Open browser to http://localhost:8501

5. End-to-End Test

Bash

```
python src/main.py \
  --resume data/resume_template.docx \
  --jd data/jd.txt \
  --out output/test_tailored.docx \
  --model local \
  --verbose
```

🚨 Troubleshooting

Common Issues and Solutions

Python/Pip Issues

Bash

```bash
# Update pip
python -m pip install --upgrade pip

# Clear pip cache
pip cache purge

# Reinstall requirements
pip uninstall -r requirements.txt -y
pip install -r requirements.txt
```

Ollama Issues

Bash

```bash
# Check if Ollama is running
curl http://localhost:11434/api/tags

# Restart Ollama service
pkill ollama
ollama serve

# Update Ollama
ollama update
```

Permission Issues (Linux/macOS)

Bash

```bash
# Fix file permissions
chmod +x src/main.py
chmod +x test_parser.py
chmod +x test_llm.py

# Fix directory permissions
chmod -R 755 resume-tailor-agent/
```

Port Conflicts

Bash

```bash
# Check what's using port 8501
lsof -i :8501  # macOS/Linux
netstat -ano | findstr :8501  # Windows

# Use different port
streamlit run streamlit_app.py --server.port 8502
```

Getting Detailed Logs

Bash

```bash
# Enable verbose logging
python src/main.py --verbose [other args]

# Check Streamlit logs
streamlit run streamlit_app.py --logger.level debug
```

🔒 Security Considerations

API Key Security

- Never commit API keys to version control
- Use environment variables or secure key management
- Rotate keys regularly
- Monitor API usage and costs

Local Security

- Keep Ollama updated

- Use virtual environments
- Limit file permissions appropriately
- Be cautious with uploaded files in web interface

Network Security

- Use HTTPS in production
- Implement proper authentication
- Consider firewall rules
- Monitor access logs

📊 Performance Optimization

Local LLM Performance

- Use SSD storage for models
- Allocate sufficient RAM
- Close unnecessary applications
- Consider GPU acceleration (if supported)

API Performance

- Monitor rate limits
- Implement retry logic
- Cache responses when appropriate
- Choose appropriate model for speed vs quality

Web Interface Performance

- Limit file upload sizes
- Implement progress indicators
- Use session state efficiently
- Consider caching for repeated operations

🔄 Updates and Maintenance

Updating Dependencies

```bash
# Update all packages
pip install --upgrade -r requirements.txt

# Update specific package
pip install --upgrade streamlit
```

Updating Ollama Models

```bash
# Update all models
ollama list
ollama pull mistral  # Update specific model
```

Backup and Recovery

- Backup your customized resumes
- Export environment configurations
- Document any custom modifications
- Keep requirements.txt updated

This setup guide should help you get the Resume Tailor Agent running in any environment. If you encounter issues not covered here, please check the main README.md or create an issue in the project repository.