SLM001 - Local Small Language Model Chatbot

A lightweight, local chatbot that runs entirely offline using a small language model. Designed to work on both AMD64 and ARM64 architectures, including NVIDIA Jetson Nano.

Features

- 🤖 **Fully Local**: Runs completely offline after initial setup
- 🏗️ **Multi-Architecture**: Supports both AMD64 and ARM64 platforms
- 💬 **Interactive Chat**: Command-line interface for real-time conversations
- 🧠 **Context Awareness**: Maintains conversation history for better responses
- 🐳 **Docker Ready**: Easy deployment with Docker containers
- ⚡ **Lightweight**: Uses DialoGPT-small model (~117MB)
- 🔒 **Privacy First**: No data sent to external servers

Quick Start

Option 1: Run with Docker (Recommended)

1. **Pull the pre-built image:**

   ```bash
   docker pull your_dockerhub_username/slm001-chatbot:latest
   ```

2. **Run the chatbot:**

   ```bash
   docker run -it --rm your_dockerhub_username/slm001-chatbot:latest
   ```

Option 2: Run Locally

1. **Clone the repository:**

   bash

   ```bash
   git clone <your-repo-url>
   cd slm001-chatbot
   ```

2. **Install dependencies:**

   bash

   ```bash
   pip install -r requirements.txt
   ```

3. **Run the chatbot:**

   bash

   ```bash
   python slm001.py
   ```

Development Setup

Prerequisites

- Python 3.9 or higher
- Docker (for containerized deployment)
- Docker Buildx (for multi-architecture builds)
- Docker Hub account (for publishing)

Local Development

1. **Test the installation:**

   bash

   ```bash
   python test_local.py
   ```

2. **Run the chatbot directly:**

   bash

   ```bash
   python slm001.py
   ```

Building for Multiple Architectures

1. **Update the build script:** Edit `build_and_deploy.sh` and replace `your_dockerhub_username` with your actual Docker Hub username.
2. **Make the script executable:**

   bash

   ```bash
   chmod +x build_and_deploy.sh
   ```

3. **Run the build script:**

   bash

   ```bash
   ./build_and_deploy.sh
   ```

This will:

- Set up Docker buildx for multi-architecture builds
- Build images for both AMD64 and ARM64
- Push to Docker Hub
- Generate deployment instructions

Usage

Interactive Commands

Once the chatbot is running, you can use these commands:

- **Chat normally**: Just type your message and press Enter
- `help`: Show available commands
- `clear`: Clear conversation history
- `quit`, `exit`, or `bye`: End the conversation

Example Conversation

```
💬 You: Hello, how are you today?
🤖 Bot: Hello! I'm doing well, thank you for asking. How can I help you today?

💬 You: What is Python programming?
🤖 Bot: Python is a high-level, interpreted programming language known for its
simplicity and readability. It's widely used for web development, data science,
automation, and artificial intelligence projects.

💬 You: quit
🤖 Bot: Goodbye! Have a great day! 👋
```

Architecture Support

This project supports the following architectures:

- **AMD64** (x86_64): Standard desktop/server processors
- **ARM64** (AArch64): ARM-based processors including:
  - NVIDIA Jetson Nano, Xavier, Orin
  - Raspberry Pi 4+ (64-bit)
  - Apple M1/M2 Macs
  - AWS Graviton instances

Jetson Nano Deployment

Requirements

- NVIDIA Jetson Nano with Ubuntu 18.04
- Docker installed and configured
- At least 2GB free RAM
- Internet connection for initial model download

Step-by-Step Deployment

1. **Ensure Docker is installed on Jetson Nano:**

   bash

   ```bash
   sudo apt update
   sudo apt install docker.io
   sudo systemctl enable docker
   sudo systemctl start docker
   sudo usermod -aG docker $USER
   ```

2. **Pull the ARM64 image:**

   bash

   ```bash
   docker pull your_dockerhub_username/slm001-chatbot:latest
   ```

3. **Run the chatbot:**

   bash

   ```bash
   docker run -it --rm your_dockerhub_username/slm001-chatbot:latest
   ```

4. **For persistent model cache:**

   bash

   ```bash
   docker run -it --rm -v slm001-cache:/app/cache your_dockerhub_username/slm001-chatb
   ```

Technical Details

Model Information

- **Base Model**: microsoft/DialoGPT-small
- **Model Size**: ~117MB
- **Parameters**: ~117M
- **Context Length**: 512 tokens
- **Language**: English
- **License**: MIT

Performance Characteristics

Platform	RAM Usage	Startup Time	Response Time
Jetson Nano	~1.5GB	~30-60s	~2-5s
Raspberry Pi 4	~1.2GB	~20-40s	~1-3s
Desktop AMD64	~1.0GB	~10-20s	~0.5-1s

System Requirements

Minimum:

- 2GB RAM
- 1GB free disk space
- ARMv8 or x86_64 CPU

Recommended:

- 4GB RAM
- 2GB free disk space
- Multi-core CPU

Customization

Using Different Models

You can modify `slm001.py` to use different models by changing the `model_name` parameter:

```python
# For a larger model (requires more RAM)
chatbot = LocalChatbot("microsoft/DialoGPT-medium")

# For a different conversation model
chatbot = LocalChatbot("microsoft/BlenderBot-400M-distill")
```

Adjusting Response Parameters

Modify these parameters in the `generate_response` method:

```python
# In the chat_pipeline call
temperature=0.7,      # Lower = more focused, Higher = more creative
max_new_tokens=100,   # Maximum response length
do_sample=True,       # Enable sampling for variety
```

Troubleshooting

Common Issues

1. **Out of Memory Error:**
   - Reduce `max_length` in the pipeline
   - Use a smaller model
   - Increase system swap space
2. **Slow Response Times:**
   - Model is running on CPU (expected for Jetson Nano)
   - First response is always slower (model loading)
   - Consider using a smaller model
3. **Docker Build Fails:**
   - Ensure Docker Buildx is installed
   - Check available disk space
   - Verify internet connection
4. **Model Download Fails:**
   - Check internet connection
   - Verify HuggingFace Hub accessibility
   - Try downloading manually first

Performance Optimization

1. **For Jetson Nano:**

   bash

   ```bash
   # Increase swap space
   sudo fallocate -l 4G /swapfile
   sudo chmod 600 /swapfile
   sudo mkswap /swapfile
   sudo swapon /swapfile
   ```

2. **For better startup time:**

   bash

   ```bash
   # Run with persistent cache
   docker run -it --rm -v slm001-cache:/app/cache your_dockerhub_username/slm001-chatb
   ```

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test on both AMD64 and ARM64 if possible
5. Submit a pull request

License

Acknowledgments

- Hugging Face Transformers for the ML framework
- Microsoft DialoGPT for the base model
- Docker for containerization support

Support

If you encounter issues:

1. Check the troubleshooting section above
2. Review the container logs: `docker logs <container-name>`
3. Test locally first: `python test_local.py`
4. Open an issue with system details and error messages

Happy Chatting! 🤖💬