SiliconFlow Toolkit

A comprehensive, enterprise-grade toolkit for integrating SiliconFlow AI models with OpenCode and Crush. Features dynamic model discovery, intelligent configuration management, and full API coverage.

✨ Features

- 🔄 **Dynamic Model Discovery**: Automatically fetches and categorizes all available SiliconFlow models
- 🎯 **Smart Configuration Merging**: Updates existing configs without breaking changes
- 🔧 **Full API Coverage**: Supports all SiliconFlow endpoints (chat, embeddings, rerank, images, audio, video)
- ✅ **Comprehensive Testing**: Real API integration tests with 97+ models validated
- 🏷️ **Schema Compatibility**: Fully compatible with OpenCode and Crush configuration schemas
- 💾 **Intelligent Caching**: Reduces API calls with smart response caching
- 🛡️ **Enterprise Security**: Secure API key handling with proper file permissions
- 📊 **Rich Logging**: Comprehensive logging and monitoring capabilities

🚀 Quick Start

Installation & Configuration

1. **Clone and install**:

```
git clone <repository-url>
cd siliconflow-toolkit
./install
```

2. **Choose installation target** (optional):

```
# Install for both OpenCode and Crush (default)
./install

# Install only for OpenCode
./install --opencode

# Install only for Crush
./install --crush

# Force reinstallation (overwrite existing configs)
./install --force
```

3. **Enter your SiliconFlow API key** when prompted

4. **Verify installation**:

```
python3 ~/.config/check_siliconflow_perf.py
```

Automatic Model Updates

Set up a cron job to keep models updated:

```
# Add to crontab -e
0 3 * * * python3 ~/.config/update_siliconflow_models.py >>
~/.siliconflow_update.log 2>&1
```

📚 Architecture

Core Components

```
siliconflow-toolkit/
├── siliconflow_client.py    # Comprehensive API client
├── model_discovery.py       # Dynamic model discovery system
├── install.py              # Enhanced configuration builder
├── test_siliconflow.py     # Comprehensive test suite
├── test_compatibility.py   # Configuration validation
└── AGENTS.md               # Agent configuration guidelines
```

API Client Features

The `SiliconFlowAPIClient` provides complete coverage of all SiliconFlow API endpoints:

Chat Completions

```
from siliconflow_client import SiliconFlowAPIClient, ChatRequest,
ChatMessage

client = SiliconFlowAPIClient("your-api-key")
response = client.chat_completion(ChatRequest(
    model="Qwen/Qwen2.5-7B-Instruct",
    messages=[ChatMessage(role="user", content="Hello!")]
))
```

Embeddings

```
```

```
from siliconflow_client import EmbeddingRequest

response = client.create_embeddings(EmbeddingRequest(
    model="text-embedding-ada-002",
    input=["Hello world", "How are you?"]
))
```

Image Generation

```
from siliconflow_client import ImageGenerationRequest

response = client.create_image(ImageGenerationRequest(
    model="black-forest-labs/FLUX.1-dev",
    prompt="A beautiful sunset over mountains"
))
```

Audio Synthesis

```
from siliconflow_client import AudioGenerationRequest

response = client.create_speech(AudioGenerationRequest(
    model="FunAudioLLM/CosyVoice2-0.5B",
    input="Hello, world!",
    voice="alloy"
))
```

Video Generation

```
from siliconflow_client import VideoGenerationRequest

response = client.create_video(VideoGenerationRequest(
    model="Wan-AI/Wan2.2-T2V-A14B",
    prompt="A cat playing in a garden",
    duration=5
))
```

Dynamic Model Discovery

The `SiliconFlowModelDiscovery` system automatically categorizes models:

```
from model_discovery import SiliconFlowModelDiscovery

discovery = SiliconFlowModelDiscovery("your-api-key")
```

```
registry = discovery.discover_all_models()

# Get category summary
summary = discovery.get_category_summary()
# Output: {'chat': {'model_count': 77, 'default_model': 'Qwen/Qwen2.5-14B-
Instruct'}, ...}
```

Configuration Builder

The `EnhancedSiliconFlowConfigBuilder` creates compatible configurations:

```
from install import EnhancedSiliconFlowConfigBuilder

builder = EnhancedSiliconFlowConfigBuilder("your-api-key")

# Generate OpenCode configuration
opencode_config = builder.build_opencode_config()

# Generate Crush configuration
crush_config = builder.build_crush_config()
```

🔧 Configuration

OpenCode Integration

The toolkit generates fully compatible OpenCode configurations:

```
{
  "$schema": "https://opencode.ai/config.json",
  "theme": "dark",
  "model": "Qwen/Qwen2.5-14B-Instruct",
  "provider": {
    "siliconflow": {
      "name": "SiliconFlow",
      "options": {
        "apiKey": "your-api-key",
        "baseURL": "https://api.siliconflow.com/v1"
      },
      "models": {
        "Qwen/Qwen2.5-7B-Instruct": {
          "name": "Qwen 2.5 7B Instruct",
          "contextWindow": 32768,
          "supportsFunctionCalling": true,
          "supportsVision": false,
          "supportsAudio": false,
          "supportsVideo": false
        }
      }
    }
```

```
    }
  }
```

Crush Integration

Compatible Crush configurations with multiple providers:

```
{
  "$schema": "https://charm.land/crush.json",
  "providers": {
    "siliconflow-chat": {
      "name": "SiliconFlow Chat Models",
      "type": "openai-compatible",
      "api_key": "your-api-key",
      "base_url": "https://api.siliconflow.com/v1/chat/completions",
      "capabilities": ["chat", "reasoning", "tool-calling", "function-
calling"],
      "models": [
        {
          "id": "Qwen/Qwen2.5-7B-Instruct",
          "name": "Qwen 2.5 7B Instruct",
          "context_window": 32768,
          "supports_mcp": true,
          "supports_lsp": true
        }
      ]
    }
  },
  "defaultProvider": "siliconflow-chat",
  "defaultModel": "Qwen/Qwen2.5-7B-Instruct"
}
```

🧪 Testing

Running Tests

```
# Run unit tests
python3 test_siliconflow.py

# Run compatibility tests
python3 test_compatibility.py

# Run with real API (requires SILICONFLOW_API_KEY)
SILICONFLOW_API_KEY=your-key python3 test_siliconflow.py

# Run full integration tests (clones and builds Crush/OpenCode locally)
SILICONFLOW_API_KEY=your-key python3 integration_test.py

# Run integration tests for specific platforms
```

```
SILICONFLOW_API_KEY=your-key python3 integration_test.py --opencode-only
SILICONFLOW_API_KEY=your-key python3 integration_test.py --crush-only

# Run integration tests with custom config paths
SILICONFLOW_API_KEY=your-key python3 integration_test.py \
  --config-home /tmp/test-config \
  --keep-temp
```

Test Coverage

- ✅ **API Client Tests**: All endpoints tested with mocked responses
- ✅ **Model Discovery Tests**: Categorization and capability inference
- ✅ **Configuration Tests**: OpenCode and Crush compatibility
- ✅ **Integration Tests**: Real API calls (when API key provided)
- ✅ **Merging Tests**: Smart configuration updates

Test Results

```
✅ Real API Integration: Successfully fetched 97 models
✅ Model Categorization: 77 chat, 12 vision, 3 audio, 2 video models
✅ Configuration Generation: 82 OpenCode models, 4 Crush providers
✅ Schema Validation: All configurations compatible
```

Integration Tests

The integration test suite provides end-to-end validation:

- **Environment Validation**: Checks API keys and required tools
- **Source Building**: Clones and builds Crush/OpenCode from GitHub
- **Configuration Installation**: Tests custom config path options
- **Headless Operation**: Starts applications in headless mode
- **Model Verification**: Validates SiliconFlow model availability
- **API Connectivity**: Tests real API calls and responses

Requirements: Git, Go 1.19+, Node.js 18+, valid `SILICONFLOW_API_KEY`

See `INTEGRATION_TESTS.md` for detailed documentation.

📊 Model Categories

Category	Count	Default Model	Capabilities
Chat	77	Qwen/Qwen2.5-14B-Instruct	Text generation, reasoning, function calling
Vision	12	Qwen/Qwen2.5-VL-7B-Instruct	Image understanding, visual chat
Audio	3	FunAudioLLM/CosyVoice2-0.5B	Speech synthesis, voice cloning
Video	2	Wan-AI/Wan2.2-T2V-A14B	Video generation from text

Category	Count	Default Model	Capabilities
Embedding	0	-	Text embeddings (coming soon)
Rerank	0	-	Document reranking (coming soon)

🔒 Security

- **API Key Protection**: Keys stored with 0o600 permissions
- **Secure Backups**: Automatic backup creation before config changes
- **No Secrets in Logs**: Sensitive data never logged
- **Environment Variables**: Support for secure key management

📈 Performance

- **Smart Caching**: 1-hour response cache reduces API calls
- **Batch Processing**: Efficient model categorization
- **Memory Optimized**: Streaming support for large responses
- **Concurrent Safety**: Thread-safe configuration updates

🐛 Troubleshooting

Common Issues

1. **API Key Not Found**

```
export SILICONFLOW_API_KEY=your-key
./install
```

2. **Permission Denied**

```
chmod +x install
chmod +x install.py
```

3. **Cache Issues**

```
rm -rf ~/.cache/siliconflow
```

4. **Configuration Conflicts**
   - The toolkit automatically merges configurations
   - Manual backup available in `~/.config/siliconflow_backups/`

Debug Mode

Enable detailed logging:

```
export SILICONFLOW_DEBUG=1
python3 install.py
```

🤝 Contributing

1. Fork the repository
2. Create a feature branch
3. Add tests for new functionality
4. Ensure all tests pass
5. Submit a pull request

Development Setup

```
# Clone the repository
git clone <repository-url>
cd siliconflow-toolkit

# Install dependencies (requests is the only external dependency)
pip install requests

# Run tests
python3 test_siliconflow.py
python3 test_compatibility.py

# Format code (using your preferred formatter)
# black . && isort .

# Check for any untracked temporary files
git status --ignored
```

File Structure

```
siliconflow-toolkit/
├── .gitignore              # Git ignore patterns for temp files
├── README.md               # Main documentation
├── API_REFERENCE.md        # Technical API documentation
├── AGENTS.md              # Agent development guidelines
├── LICENSE                 # MIT license
├── install.py             # Main installation script
├── siliconflow_client.py  # API client implementation
├── model_discovery.py     # Dynamic model discovery
├── test_siliconflow.py    # Unit tests
├── test_compatibility.py  # Compatibility tests
├── fix_crush_config.py    # Configuration fixer
├── sync.sh                # Config synchronization
├── update_tools.sh        # Tool updates
└── install.sh             # Installation wrapper
```

📄 License

MIT License - see LICENSE for details.

🙏 Acknowledgments

- **SiliconFlow**: For providing comprehensive AI infrastructure
- **Charm**: For OpenCode and Crush platforms
- **Community**: For model contributions and testing

🔧 Troubleshooting

Models Not Appearing in Crush

If you see only reranker and embedding models in Crush:

1. **Check Configuration**:

   ```
   python3 fix_siliconflow_config.py
   ```

2. **Restart Crush**: After fixing the configuration, restart Crush completely

3. **Verify Environment Variables**:

   ```
   source ~/.config/siliconflow_env.sh
   echo $OPENAI_API_KEY
   echo $OPENAI_BASE_URL
   ```

4. **Check Configuration File**:

   ```
   cat ~/.config/crush/crush.json | jq '.providers["siliconflow-
   chat"].models | length'
   ```

Common Issues

- **Missing Models**: Run the comprehensive fix script
- **API Key Issues**: Ensure SILICONFLOW_API_KEY is set and valid
- **Permission Issues**: Check file permissions on crush.json (should be 600)

📞 Support

- **Issues**: GitHub Issues
- **Discussions**: GitHub Discussions
- **Discord**: Join our community

Built with ❤ for the AI development community