Ollama Local LLM Guide

This guide provides an overview of working with Ollama's local LLM models, based on our testing and findings.

Available Models

The following models are available on your local Ollama instance:

Model	Parameters	Quantization
qwen2.5-coder:latest	7.6B	Q4_K_M
deepseek-r1:32b	32.8B	Q4_K_M
deepseek-r1:1.5b	1.8B	Q4_K_M
deepseek-r1:latest	7.6B	Q4_K_M
llama3.3:latest	70.6B	Q4_K_M
nchapman/dolphin3.0-qwen2.5:latest	3.1B	Q4_K_M
qwen2.5:latest	7.6B	Q4_K_M
phi4:latest	14.7B	Q4_K_M
dolphin3:latest	8.0B	Q4_K_M

API Workflow Diagrams

Basic API Request Flow

```
flowchart LR
    A[Client] -->|HTTP Request| B[Ollama API]
    B -->|JSON Response| A
    B <-->|Model Loading| C[(Local Models)]

    style A fill:#2a4b8d,stroke:#333,stroke-width:2px,color:#ffffff
    style B fill:#006699,stroke:#333,stroke-width:2px,color:#ffffff
    style C fill:#217645,stroke:#333,stroke-width:2px,color:#ffffff
```

Generate API vs Chat API

```
flowchart TD
    A[Client] --> B{API Choice}
    B -->|Generate API| C[No Context Preservation]
    B -->|Chat API| D[Context Preservation]
    C -->|New Request| E[Include Full Context]
    C -->|Or| F[No Context]

    D -->|Messages Array| G[Previous Messages + New Message]

    E --> H[Model Response]
    F --> H
    G --> H

    style A fill:#2a4b8d,stroke:#333,stroke-width:2px,color:#ffffff
    style B fill:#006699,stroke:#333,stroke-width:2px,color:#ffffff
    style C fill:#a63232,stroke:#333,stroke-width:2px,color:#ffffff
    style D fill:#217645,stroke:#333,stroke-width:2px,color:#ffffff
    style E fill:#7d5700,stroke:#333,stroke-width:2px,color:#ffffff
    style F fill:#7d5700,stroke:#333,stroke-width:2px,color:#ffffff
    style G fill:#7d5700,stroke:#333,stroke-width:2px,color:#ffffff
    style H fill:#217645,stroke:#333,stroke-width:2px,color:#ffffff
```

Model Size vs. Accuracy

```
graph LR
    A[Model Size] -->|Increases| B[Accuracy]
    A -->|Decreases| C[Speed]
    A -->|Increases| D[Memory Usage]
    A -->|Decreases| E[Hallucination Risk]

    style A fill:#006699,stroke:#333,stroke-width:2px,color:#ffffff
    style B fill:#217645,stroke:#333,stroke-width:2px,color:#ffffff
    style C fill:#a63232,stroke:#333,stroke-width:2px,color:#ffffff
    style D fill:#a63232,stroke:#333,stroke-width:2px,color:#ffffff
    style E fill:#217645,stroke:#333,stroke-width:2px,color:#ffffff
```

API Usage Examples

List Available Models

```
curl -s http://localhost:11434/api/tags
```

Basic Generate API Request

```
curl -s http://localhost:11434/api/generate -d '{
  "model": "deepseek-r1:1.5b",
  "prompt": "Why is Studio Ghibli art so popular?",
  "stream": false
}'
```

Chat API with Context

```
curl -s http://localhost:11434/api/chat -d '{
  "model": "deepseek-r1:32b",
  "stream": false,
  "messages": [
    {"role": "user", "content": "Why is Studio Ghibli art so popular?"},
    {"role": "assistant", "content": "Studio Ghibli art has gained
unparalleled popularity due to its rich historical and cultural
heritage..."},
    {"role": "user", "content": "Tell me more about Spirited Away"}
  ]
}'
```

Model Comparison

Hallucination Test Results

We tested the same query about "Spirited Away" on different model sizes and found:

```
graph TD
    A[Query about Spirited Away] --> B[deepseek-r1:1.5b]
    A --> C[deepseek-r1:32b]

    B --> D[Hallucinations]
    C --> E[Accurate Information]

    D --> D1[Called it My Hero Academia]
    D --> D2[Said it was from 1968]
    D --> D3[Invented fake characters]
    D --> D4[Called it a short film]

    E --> E1[Correct Japanese title]
    E --> E2[Accurate plot description]
    E --> E3[Real characters mentioned]
    E --> E4[Correct release date 2001]
    E --> E5[Academy Award details]

    style A fill:#2a4b8d,stroke:#333,stroke-width:2px,color:#ffffff
    style B fill:#a63232,stroke:#333,stroke-width:2px,color:#ffffff
    style C fill:#217645,stroke:#333,stroke-width:2px,color:#ffffff
    style D fill:#a63232,stroke:#333,stroke-width:2px,color:#ffffff
    style E fill:#217645,stroke:#333,stroke-width:2px,color:#ffffff
    style D1 fill:#a63232,stroke:#333,stroke-width:1px,color:#ffffff
    style D2 fill:#a63232,stroke:#333,stroke-width:1px,color:#ffffff
    style D3 fill:#a63232,stroke:#333,stroke-width:1px,color:#ffffff
    style D4 fill:#a63232,stroke:#333,stroke-width:1px,color:#ffffff
    style E1 fill:#217645,stroke:#333,stroke-width:1px,color:#ffffff
    style E2 fill:#217645,stroke:#333,stroke-width:1px,color:#ffffff
    style E3 fill:#217645,stroke:#333,stroke-width:1px,color:#ffffff
    style E4 fill:#217645,stroke:#333,stroke-width:1px,color:#ffffff
    style E5 fill:#217645,stroke:#333,stroke-width:1px,color:#ffffff
```

Best Practices

1. **Use the chat API format** with message history for conversational interactions
2. **Include previous context** in the messages array
3. **Prefer larger models** when accuracy is important
4. **Set stream to false** for easier parsing of complete responses
5. **Consider model size tradeoffs** - larger models are more accurate but slower and use more memory

Performance Considerations

```
quadrantChart
    title Model Performance Quadrant
    x-axis Low Memory Usage --> High Memory Usage
    y-axis Low Accuracy --> High Accuracy
    quadrant-1 Fast but less accurate
    quadrant-2 Ideal for most use cases
    quadrant-3 Limited usefulness
    quadrant-4 Accurate but resource-heavy
    "deepseek-r1:1.5b": [0.2, 0.3]
    "deepseek-r1:latest": [0.5, 0.6]
    "qwen2.5:latest": [0.5, 0.65]
    "phi4:latest": [0.7, 0.75]
    "deepseek-r1:32b": [0.8, 0.85]
    "llama3.3:latest": [0.95, 0.9]
```

Conclusion

Local Ollama models provide a powerful way to run LLMs on your own hardware. By understanding the tradeoffs between model size, accuracy, and performance, you can choose the right model for your specific needs. The API design allows for both simple one-off queries and more complex conversational interactions.

Thoughts from Claude 3.7 Sonnet

As Claude 3.7 Sonnet powering the Windsurf Cascade agent from Codeium, I have some perspectives on the comparison between local LLMs like those in Ollama and cloud-based models like myself:

Advantages of Local LLMs

```
mindmap
  root((Local LLMs))
    Privacy
      No data leaves your device
      Complete control over usage
    Latency
      No network dependency
      Consistent response times
    Offline Access
      Works without internet
      Resilient to outages
    Cost
      No usage-based billing
      One-time download
```

When Cloud Models May Be Preferable

While local models offer significant advantages, there are scenarios where cloud models like myself might be more suitable:

1. **When accuracy is paramount** - Larger cloud models (100B+ parameters) can provide more factual, nuanced responses with less hallucination
2. **For specialized tasks** - Models with domain-specific training or fine-tuning
3. **When local hardware is limited** - Running 70B+ models locally requires significant GPU resources
4. **For integration with other services** - Cloud APIs often offer better integration capabilities
5. **When freshness of knowledge matters** - Cloud models can be updated more frequently with new information

The Hybrid Approach

In my view, the ideal setup for many users combines both approaches:

- Use local models for sensitive data, quick queries, and when offline
- Use cloud models for complex reasoning, specialized knowledge, or when highest accuracy is needed
- Consider privacy implications and choose the appropriate model for each specific use case

The testing we did today demonstrates both the capabilities and limitations of local models. The hallucination example with the smaller deepseek model shows why model size matters, while the accurate response from the larger model shows how close local LLMs are getting to cloud quality.

As local hardware and model efficiency continue to improve, the gap between local and cloud models will likely narrow further, giving users even more powerful options for running AI locally.