prepared by Mayuresh Ratnaparkhi

LLM Tools Comparison with llama.cpp Spring Boot Integration

LLM Tools Comparison Matrix

Tool	Speed (CPU)	Speed (GPU)	Easy Setup	Java API Integration	Comment
Ollama	Medium	Medium	✅ **Easy**	✅ **(Spring AI)**	Good balance of features
llama.cpp	✅ **Fast**	✅ **Very Fast**	⚠️ CLI/C++	✅ **via HTTP/Java native**	Fastest on CPU (quantized)
vLLM	❌ Slow	✅✅ **Blazing**	⚠️ Medium	✅ **(OpenAI API)**	Best for GPU batch inferencing
FastChat	Medium	Fast	⚠️ Medium	✅ **(OpenAI API)**	Full chat server stack
OpenLLM	Medium	Fast	⚠️ Medium	✅ **(REST/gRPC)**	Highly customizable

> **Key Insight:** For CPU-based local development, `llama.cpp` with quantized models provides the best performance-to-resource ratio.

llama.cpp with Spring Boot Integration

Fastest on CPU (local/dev): llama.cpp via llama-cpp-java (Java bindings)

To use llama.cpp with Spring Boot, the most practical approach is to run llama.cpp as a local HTTP server and then call it from Spring Boot using a REST client.

Prepared by Mayuresh Ratnaparkhi

Step-by-Step Implementation Guide

1 Install Dependencies

```
sudo apt update
sudo apt install cmake build-essential pkg-config libopenblas-dev
```

2 Install libcurl Development Package

```
sudo apt install libcurl4-openssl-dev
```

For GPU Support: Install CUDA toolkit if you have NVIDIA GPU

```
sudo apt install nvidia-cuda-toolkit
```

3 Clone llama.cpp and Build

```
git clone https://github.com/ggerganov/llama.cpp
cd llama.cpp
mkdir build
cd build
cmake ..
cmake --build . --config Release
```

4 Download Quantized Model

```
cd llama.cpp
wget https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.1-GGUF
```

> **Tip:** The Q4_K_M quantization provides good balance between quality and performance

5 Enable Server Mode

```
cd llama.cpp/build
cmake .. -DLLAMA_BUILD_SERVER=ON
cmake --build . --config Release
```

6 Start the LLM HTTP Server

```
./bin/server -m ../models/mistral-7b-instruct-v0.1.Q4_K_M.gguf --p
```

> **Expected Output:** Server should start and show available endpoints

Spring Boot Integration

7 Add RestTemplate Configuration

```
@Configuration
public class AppConfig {
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
```

```
    }
}
```


8 Create LLM Service

```
@Service
public class LlamaService {
    private final RestTemplate restTemplate;

    public LlamaService(RestTemplate restTemplate) {
        this.restTemplate = restTemplate;
    }

    public String generateResponse(String prompt) {
        String url = "http://localhost:8000/completion";

        Map<String, Object> request = new HashMap<>();
        request.put("prompt", prompt);
        request.put("n_predict", 100);

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);

        HttpEntity<Map<String, Object>> entity = new HttpEntity<>(

        ResponseEntity<Map> response = restTemplate.postForEntity(
            url, entity, Map.class);

        return response.getBody().get("content").toString();
    }
}
```

9 Create REST Controller

```
@RestController
@RequestMapping("/api/llm")
public class LlamaController {
    private final LlamaService llamaService;

    public LlamaController(LlamaService llamaService) {
        this.llamaService = llamaService;
    }

    @PostMapping("/ask")
    public String askQuestion(@RequestBody String prompt) {
        return llamaService.generateResponse(prompt);
    }
}
```

Testing the Integration

10 **Test with cURL**

```
curl -X POST http://localhost:8080/api/llm/ask \
  -H "Content-Type: text/plain" \
  -d "What is Spring Boot?"
```

Performance Tip: For production use, consider:

- Adding connection pooling
- Implementing timeout handling
- Adding authentication to the llama.cpp server

Alternative: Java Native Bindings

For even better performance, consider using the `llama-cpp-java` bindings:

```
// Add to pom.xml
<dependency>
    <groupId>com.github.llama-cpp</groupId>
    <artifactId>llama-cpp-java</artifactId>
    <version>1.0.0</version>
</dependency>

// Usage example
LlamaModel model = new LlamaModel("models/mistral-7b-instruct-v0.1.Q4
String response = model.generate("What is Java?");
```