Tool Use and Function Calling

Week 3: Extending LLMs with External Capabilities

PhD Course in Agentic Artificial Intelligence

Learning Objectives

Bloom's Taxonomy Levels

- **Remember**: Define function calling, MCP protocol, tool augmentation
- **Understand**: Explain how LLMs learn to use external tools
- **Apply**: Implement tool-using agents with OpenAI and Anthropic APIs
- **Analyze**: Compare different tool integration approaches
- **Evaluate**: Assess tool selection strategies and error handling
- **Create**: Design custom tools and integrate into agent workflows

Tool use transforms LLMs from text generators into action-taking agents.

Why Tools Matter

LLM Limitations

- Knowledge cutoff (can't access current information)
- No computation ability (unreliable at math)
- Can't interact with external systems
- Hallucinate when uncertain

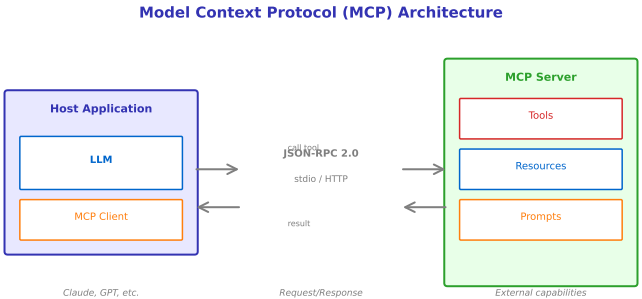
Tools Enable

- Real-time information retrieval (web search, APIs)
- Precise computation (calculators, code execution)
- System interaction (databases, file systems)
- Grounded responses (verified facts)

Toolf

(Schick et al., 2023): LLMs can learn to use tools autonomously.

Model Context Protocol (MCP)



MCP

is Anthropic's open protocol for connecting LLMs to external tools.

MCP Core Concepts

Three Primitives

- **Tools**: Functions the LLM can call (search, calculate, etc.)
- **Resources**: Data sources (files, databases, APIs)
- **Prompts**: Reusable prompt templates

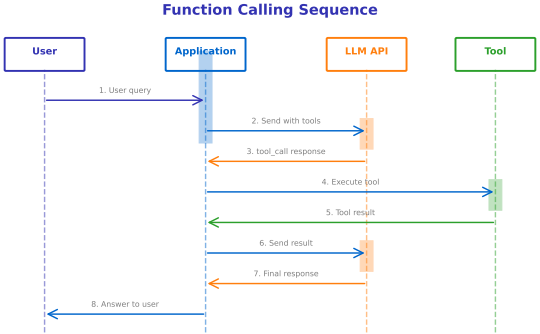
Transport Layer

- JSON-RPC 2.0 over stdio (local) or HTTP (remote)
- Request/response pattern with typed schemas
- Supports streaming for long-running operations

Key Advantage: Standardized protocol across different LLM providers

MCP enables tool portability: write once, use with any LLM.

Function Calling Sequence



The LLM decides when and which tool to call based on the query.

OpenAI Function Calling

Tool Definition (JSON Schema)

```
{"type":"function","function":{"name":"get_weather",
 "description":"Get weather for a location",
 "parameters":{"type":"object",
  "properties":{"location":{"type":"string"}},
  "required":["location"]}}}
```

Key Points

- Clear descriptions help LLM choose correct tool
- JSON Schema for parameter validation
- LLM returns structured `tool_calls` array

Good tool descriptions are critical for reliable tool selection.

Anthropic Tool Use

Claude Tool Definition

```
{"name":"get_weather",
 "description":"Get weather for a location",
 "input_schema":{"type":"object",
  "properties":{"location":{"type":"string"}},
  "required":["location"]}}
```

Differences from OpenAI

- Uses `input_schema` instead of `parameters`
- Returns `tool_use` content blocks
- Supports `tool_choice` for forcing specific tools

and OpenAI have similar but not identical APIs.

Tool Selection Strategies

Selection Approaches

- **Auto**: LLM decides when/which tool (default)
- **Required**: Force at least one tool call
- **Specific**: Force a particular tool
- **None**: Disable tool use

Best Practices

- Start with “auto” and monitor selection quality
- Use “required” when you know a tool is needed
- Provide 3-5 tools max for reliable selection
- Include negative examples in descriptions

Too many tools degrade selection accuracy – keep toolsets focused.

Error Handling

Common Failure Modes

- Tool not found or invalid parameters
- Tool execution timeout
- External API failure
- Malformed tool response

Recovery Strategies

- **Retry with backoff**: For transient failures
- **Fallback tools**: Alternative tools for same task
- **Graceful degradation**: Inform user of limitation
- **Self-correction**: Let LLM try different approach

Robu

error handling is essential for production agents.

API-Bank and ToolLLM

API-Bank (Li et al., 2023)

- Benchmark with 73 APIs across 3 levels of complexity
- Tests tool selection, parameter extraction, composition
- GPT-4 achieves 80%+ on single-tool tasks

ToolLLM (Qin et al., 2024)

- 16,000+ real-world APIs from RapidAPI
- Trains open models for tool use
- Introduces ToolBench for evaluation

Gorilla (Patil et al., 2023)

- LLM fine-tuned for API documentation
- Reduces hallucination in API parameters

Resea

shows tool use is learnable but challenging to scale.

Security Considerations

Risks

- **Prompt injection**: User tricks LLM into calling dangerous tools
- **Parameter injection**: Malicious inputs to tool parameters
- **Privilege escalation**: Tools with more access than intended
- **Data exfiltration**: Tools sending sensitive data externally

Mitigations

- Validate and sanitize all tool inputs
- Implement principle of least privilege
- Use allowlists for external connections
- Log all tool calls for audit

Tool use introduces new attack surfaces – security is critical.

Building Custom Tools

Tool Design Checklist

1. Clear, specific name (verb + noun)
2. Detailed description with examples
3. Minimal required parameters
4. Typed parameters with constraints
5. Structured return format

Example: Search Tool

- Name: `search_web`
- Description: "Search the web for current information. Use for recent events, facts to verify, or when knowledge might be outdated."
- Parameters: `query` (string, required)
- Returns: structured JSON with title, snippet, url

Good tool design = clear intent + minimal friction.

Required Readings

This Week

- Anthropic MCP Documentation: `docs.anthropic.com/mcp`

Supplementary

- Schick et al. (2023). "Toolformer." arXiv:2302.04761
- Li et al. (2023). "API-Bank." arXiv:2304.08244
- Qin et al. (2024). "ToolLLM." arXiv:2307.16789
- Patil et al. (2023). "Gorilla." arXiv:2305.15334

MCP documentation is required; papers provide research context.

Summary and Key Takeaways

Key Concepts

- **Function Calling**: Structured tool invocation by LLMs
- **MCP**: Open protocol for tool integration
- **Tool Design**: Clear names, descriptions, typed parameters
- **Security**: Validate inputs, minimize privileges

Key APIs

- OpenAI: `tools` parameter with JSON Schema
- Anthropic: `tools` with `input_schema`

Next Week

- Planning and Reasoning
- Reflexion: Learning from mistakes

transform LLMs from passive responders to active agents.