Task 09: Understanding Tool Calling / Function Calling in LLMs

❖ What is Tool Calling / Function Calling?

Tool Calling (also known as **Function Calling**) allows Large Language Models (LLMs) to **invoke external tools, functions, or APIs** when needed to complete a user's query more accurately.

Instead of just generating text, the LLM can say:

"I don't know the answer directly, but I know which function/tool to call to get it!"

❖ Why is Tool Calling Important?

- Enhances **reasoning and problem-solving**
- Allows **real-time information retrieval** (e.g., weather, live data)
- Integrates **LLMs with real-world software & APIs**
- Enables **hybrid AI systems** (LLMs + Tools)

❖ How Does It Work?

1. **Schema Definition**
   Developer defines tool/function schemas (name, parameters, types).
2. **User Query**
   User asks a question (e.g., "What's the weather in Karachi?")
3. **Model Decision**
   LLM analyzes the query and determines which tool/function to call.
4. **Tool Execution**
   The specified function/tool is executed with given parameters.
5. **LLM Response**
   LLM uses the tool's output to generate a final, accurate response.

➢ Example:

```
User: "Convert 500 PKR to USD"

LLM calls:
{
  "function": "convertCurrency",
  "parameters": {
    "from": "PKR",
    "to": "USD",
    "amount": 500
  }
}

Tool returns: 1.78 USD
LLM says: "500 PKR is approximately 1.78 USD."
```

❖ Where is Tool Calling Used?

- **Chatbots & AI Assistants**
  e.g., ChatGPT with plugins, custom tools
- **Customer Support Bots**
  e.g., checking order status
- **E-commerce Assistants**
  e.g., filtering products
- **Finance Tools**
  e.g., currency conversion, stock prices
- **Healthcare Assistants**
  e.g., symptom checker

❖ Benefits

- More accurate, dynamic responses
- Bridge between LLM and real-world data
- Safer — avoids hallucination by relying on factual tools
- Can chain multiple tools together for complex tasks

❖ Challenges

- **Security**: Tools must be validated to avoid misuse
- **Latency**: Depends on tool/API response time
- **Complexity**: Requires good design of function schemas
- ✕ **Fallbacks**: Model must handle tool errors properly

❖ Future of Tool Calling

- Seamless integration with **databases, APIs, devices**
- Powerful **AI agents** capable of multi-step planning
- Integration with **IoT**, **AR/VR**, **voice assistants**, and **robotics**

➢ **Tool calling turns LLMs from smart text generators into intelligent agents capable of interacting with the world.**

Prepared by Areeba Zafar