Tópicos Especiais em Engenharia de Software e Sistemas Computacionais I

Projeto de final de curso

Prof. Eduardo Guerra e Prof. Gabriel Gadelha

Objetivo

O objetivo deste projeto é avaliar, de forma sistemática, como diferentes modelos LLM lidam com APIs que expõem TOOLs, considerando variações na definição das operações. Especificamente, pretende-se comparar o comportamento dos modelos ao interagirem com APIs que oferecem as mesmas funcionalidades, mas com estruturas de métodos distintas.

Tarefa

Cada aluno receberá um modelo específico disponível na ferramenta Ollama. Com base no modelo, você deverá:

1. Instalar o Ollama e baixar o modelo atribuído.
2. Implementar o código necessário nas classes correspondentes às TOOLs, simulando o retorno das chamadas de cada método e registrando quais métodos foram invocados (por exemplo, por meio de logs ou contadores).
3. Implementar o código em Java, utilizando o framework LangChain4J para executar os prompts e fornecer as ferramentas (TOOLs) conforme indicado.
4. Definir os critérios de aceitação para cada prompt e cenário (conforme descrito à frente).
5. Executar os prompts deste documento com as diferentes configurações de TOOLs.
6. Coletar e analisar os resultados conforme os critérios especificados na seção Resultados e Medições.

Os alunos podem trocar informações sobre dúvidas de implementação, mas cada um deve realizar suas próprias execuções com o modelo que lhe foi atribuído e gerar seu relatório.

As APIs comparadas são:

Considere as seguintes duas APIs para comparação. Elas utilizam as anotações para a definição de Tool no framework Lanchain4J.

```java
public class BankToolsA {

	@Tool("Withdraw a value from an account "
			+ "and return if the operation was successfull or not")
	public boolean withdraw(
			@P("account number") String accountNumber,
			@P("value to be withdraw") double value) {
		//implementation omitted
	}

	@Tool("Deposit the value into an account "
			+ "and return if the operation was successfull or not")
	public boolean deposit(
			@P("account number") String accountNumber,
			@P("value to be deposited") double value) {
		//implementation omitted
	}

	@Tool("Perform a payment with a value using the money from an account "
			+ "and return if the operation was successfull or not")
	public boolean payment(
			@P("account number") String accountNumber,
			@P("value of the payment") double value) {
		//implementation omitted
	}

	@Tool("Charge the value of a tax from the account "
			+ "and return if the operation was successfull or not")
	public boolean taxes(
			@P("account number") String accountNumber,
			@P("value of the tax") double value) {
		//implementation omitted
	}

	@Tool("Return a value of a failed operation to an account "
			+ "and return if the operation was successfull or not")
	public boolean returnValue(
			@P("account number") String accountNumber,
			@P("value of the tax") double value) {
		//implementation omitted
	}
}
```

```java
public class BankToolsB {

	@Tool("Execute an operation in an account with a given value "
			+ "and return if the operation was successfull or not")
	public boolean executeOperation(
		@P("WITHDRAW if the operation is a withdraw,"
			+ " DEPOSIT if the operation is a deposit,"
			+ " TAX to charge the value of a tax from an account,"
			+ " RETURN to return the value of a failed operation,"
			+ " PAYMENT to perform a payment") OperationType type,
			@P("account number") String accountNumber,
			@P("value to be used in the operation") double value) {
		//implementation omitted
	}
}

//defined in a different file
public enum OperationType {
	WITHDRAW, DEPOSIT, TAX, RETURN, PAYMENT
}
```

Observe que as duas APIs oferecem as mesmas operações: na primeira, são apresentados cinco métodos distintos; na segunda, apenas um, no qual o tipo de operação é passado como parâmetro.

Configuração de TOOLS

Deve-se comparar as seguintes configurações de TOOLS utilizando os prompts e cenários descritos a seguir:

- CONF1: BankToolsA
- CONF2: BankToolsB
- CONF3: BankToolsA, BankToolsB
- CONF4: BankToolsB, BankToolsA

Prompts e cenários

A seguir, estão descritos os 3 prompts que deverão ser executados com acesso às operações dos TOOLS. Para cada prompt, existem dois cenários que dependem das respostas dos métodos invocados.

PROMPT1: *Transfer 1000 from account BC12345 to the account ND87632 by withdrawing from the first and depositing into the second. If both operations are successful, change 1.50 from the first account. If not, return the value to the account and don't charge the tax.*

- SCENARIO1A: Ambas as operações, saque e depósito, são executadas com sucesso.

- SCENARIO1B: A operação de saque é executada com sucesso, mas a de depósito falha.

PROMPT2: *Execute withdrawal operations of 500 from account BC3456A one at a time. Repeat until a failure is received, or until this operation has been executed 5 times. Deposit the total value withdrawn in account FG62495S and pay a tax of 10% of the value deposited in the account FG62495S.*

- SCENARIO2A: Todas as operações de saque são executadas com sucesso.
- SCENARIO2B: As três primeiras operações de saque são executadas com sucesso, e quaisquer outras realizadas depois falham.

PROMPT3: *Withdraw 600 from account AG7340H and 700 from account TG23986Q. If one of the operations is not successful, return the value to the other account and don't execute anything else. If both operations are successful, perform a deposit of the summed value into account WS2754T and perform a payment of 1200 in this same account.*

- SCENARIO3A: As duas operações de saque são executadas com sucesso.
- SCENARIO3B: Uma operação de saque é executada com sucesso e a outra não.

Definição de Critérios de Aceitação

Antes de iniciar as execuções, **cada aluno deve definir os critérios de aceitação esperados** para cada combinação de *prompt* e *cenário* (A e B).

Esses critérios de aceitação devem descrever, **de forma textual ou em formato de tabela**, o comportamento esperado do sistema, incluindo:

- **Que métodos** devem ser invocados em cada etapa.
- **A sequência esperada das chamadas.**
- **Os parâmetros esperados** para cada invocação.
- **Que operações não devem ocorrer** nos casos de falha parcial (como nos cenários B).

O objetivo é permitir a comparação entre o comportamento esperado e o efetivamente observado durante as execuções dos modelos.

A tabela a seguir apresenta um exemplo da definição do critério de aceitação do primeiro cenário do primeiro prompt.

Prompt	***Cenário***	***Operações***
1	A	withdraw(BC12345, 1000) deposit(ND87632, 1000) taxes(BC12345, 1.50)

Todos os alunos da turma devem concordar com os mesmos critérios de aceitação, mas eles devem constar nos relatórios de cada um.

Resultados e medições

Para cada combinação de configuração, prompt e cenário, devem-se realizar 10 execuções. Os seguintes dados devem ser obtidos:

- **Corretude:** A execução foi feita corretamente? Os métodos foram invocados corretamente, com os parâmetros corretos?
- **Consistência**: Todas as execuções na mesma configuração, prompt e cenário tiveram o mesmo resultado?
- **Abordagem**: Para os cenários em que as duas TOOLs foram passadas como parâmetros, qual foi invocada pelo modelo?

Nos casos em que a invocação não foi correta e não houve consistência, devem ser detalhados os fatos ocorridos.

Entrega

Cada aluno deve entregar o código criado em um arquivo .zip, bem como os critérios de aceitação definidos e os resultados da execução em um relatório.

Modelos
Cada aluno deve realizar a atividade utilizando o modelo que lhe foi atribuído abaixo:

deepseek-r1:latest -> ERIC DIEGO
qwen3:latest -> FRANCISCO WILKINIS
llama3.2:lastest -> JOSÉ ERILDO
mistral:latest -> GABRIEL BATISTUTA
gemma2:latest -> FRANCISCO ADAM
llama2:latest -> JOÃO PAULO

dolphin3:latest -> BERNARDO JOSÉ
gpt4-mini (OpenAPI) -> VICTORIA ÍRIS
gpt4 (OpenAPI) -> LUCAS BRUNO
gemini (via API) -> THIAGO TORRES

Em caso de dúvidas, enviem email para: eduardo.guerra@unibz.it