dgroove
Tecnologia emozionale.

ICT Training Center

Il tuo partner per la Formazione e la Trasformazione digitale della tua azienda

Note

Spring AI

Generative Artificial Intelligence con Java

Simone Scannapieco

Corso avanzato per Venis S.p.A, Venezia, Italia

Novembre 2025

Note

STEP-BACK PROMPTING

Note

1. *Step-back prompting*
   - Strategia: attivare parti di conoscenza di *background* prima di utilizzare il LLM per la *task*
     - Primo *prompt* per chiedere informazioni generali correlate alla *task* finale
     - Secondo *prompt* per sfruttare tali informazioni come contesto specifico e risolvere la *task* finale
     - ⚠ Focalizzare l'attenzione del LLM su parti della sua conoscenza
   - Vantaggi
     - Utente lavora molto poco sulla creazione del contesto
     - Contesti generalmente più mirati (generati dallo stesso LLM)
     - Tende a mitigare i *bias* cognitivi del LLM
   - Svantaggi
     - Costoso (utilizzo di due meccanismi di *prompting*)

Note

- Sistema di generazione nuova ambientazione per artefatto e genere
  1. Modifica modello `Artifact.java` per inserimento caratteristica genere
  2. Creazione *string template* per ambientazioni chiave nuovo artefatto (*system* e *user*)
  3. Creazione *string template* per generazione artefatto da lista di ambientazioni chiave
  4. Modifiche ad interfaccia ed implementazione del servizio Gemini
  5. Modifica del controllore MVC per servizio Gemini
  6. *Test* delle funzionalità con Postman/Insomnia

Note

Modello per artefatto

```java
package it.venis.ai.spring.demo.model;

import java.util.Objects;
import com.fasterxml.jackson.annotation.JsonPropertyDescription;
import it.venis.ai.spring.demo.data.ArtifactType;

public record Artifact(@JsonPropertyDescription("Il titolo dell'opera") String title,
        @JsonPropertyDescription("Il sottotitolo dell'opera") String subtitle,
        @JsonPropertyDescription("Il tipo dell'opera") ArtifactType type,
        @JsonPropertyDescription("Il genere dell'opera") String genre,
        @JsonPropertyDescription("La trama o la recensione dell'opera") String body) {

    @Override
    public String title() {
        return Objects.requireNonNullElse(this.title, "---");
    }

    @Override
    public String subtitle() {
        return Objects.requireNonNullElse(this.subtitle, "---");
    }

    @Override
    public String genre() {
        return Objects.requireNonNullElse(this.genre, "non specificato");
    }

}
```


Note

File `get-key-settings-for-artifact-system-prompt.st`

```
Sei un agente con accesso alla più grande base dati di opere di categoria
{artefatto} e genere {genere}.
Rispondi in modo puntuale e senza formule di introduzione.
```

File `get-key-settings-for-artifact-user-prompt.st`

```
Crea un elenco puntato di {numero} ambientazioni chiave per scrivere una trama
avvincente e di successo.
```


Note

File `get-generated-artifact-prompt.st`

```
Contesto: {lista}
Scegli una delle ambientazioni e scrivi in {numero} paragrafo/i la trama di un
nuovo {artefatto} in materia di {genere} con una trama avvincente.
{formato}
Traduci titolo, sottotitolo, genere e corpo della risposta in lingua italiana.
```


Note

Interfaccia servizio Gemini

```java
package it.venis.ai.spring.demo.services;

import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.ArtifactRequest;
import it.venis.ai.spring.demo.model.DefinitionRequest;
import it.venis.ai.spring.demo.model.DefinitionResponse;
import it.venis.ai.spring.demo.model.Question;

public interface GeminiFromClientService {

    String getAnswerFromClient(String question);

    Answer getAnswerFromClient(Question question);

    Answer getDefinitionFromClient(DefinitionRequest definitionRequest);

    Answer getCustomFormatDefinitionFromClient(DefinitionRequest definitionRequest);

    Answer getJSONUserFormatDefinitionFromClient(DefinitionRequest definitionRequest);

    DefinitionResponse getJSONOutputConverterFormatDefinitionFromClient(DefinitionRequest definitionRequest);

    Answer getSentimentForArtifact(ArtifactRequest artifactRequest);

    Answer getNERinYAMLForArtifact(ArtifactRequest artifactRequest);

    Answer getSuggestionForArtifact(ArtifactRequest artifactRequest);

    Artifact getGeneratedArtifact(ArtifactRequest artifactRequest, Integer numChoices, Integer numParagraphs);

}
```


Note

Implementazione servizio Gemini - I

```
package it.venis.ai.spring.demo.services;

import java.util.Map;
import org.springframework.ai.chat.client.ChatClient;
import org.springframework.ai.template.st.StTemplateRenderer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.Resource;
import org.springframework.stereotype.Service;
import it.venis.ai.spring.demo.model.Answer;
import it.venis.ai.spring.demo.model.DefinitionRequest;
import it.venis.ai.spring.demo.model.Question;

@Service
public class GeminiFromClientServiceImpl implements GeminiFromClientService {

    private final ChatClient chatClient;

    public GeminiFromClientServiceImpl(ChatClient.Builder chatClientBuilder) {
        this.chatClient = chatClientBuilder.build();
    }

    @Override
    public String getAnswerFromClient(String question) {
        return this.chatClient.prompt()
                .user(question)
                .call()
                .content();
    }
    ...
```


Note

Implementazione servizio Gemini - II

```
    ...
    @Value("classpath:templates/get-key-settings-for-artifact-system-prompt.st")
    private Resource keySettingsForArtifactSystemPrompt;

    @Value("classpath:templates/get-key-settings-for-artifact-user-prompt.st")
    private Resource keySettingsForArtifactUserPrompt;

    @Value("classpath:templates/get-generated-artifact-prompt.st")
    private Resource generatedArtifactPrompt;

    @Override
    public Artifact getGeneratedArtifact(ArtifactRequest artifactRequest, Integer numChoices, Integer numParagraphs) {
        BeanOutputConverter<Artifact> converter = new BeanOutputConverter<>(Artifact.class);
        String listResponse = this.chatClient.prompt()
                .options(ChatOptions.builder()
                .model("gemini-2.0-flash")
                .temperature(1.0)
                //.topP(1.0)
                //.topK(30)
                .maxTokens(2000)
                //.frequencyPenalty(0.1)
                //.presencePenalty(0.1)
                .build())
                .system(s -> s.text(this.keySettingsForArtifactSystemPrompt)
                        .params(Map.of("artefatto", artifactRequest.artifact().type().getArtifactType(),
                                       "genere", artifactRequest.artifact().genre())))
                .user(u -> u.text(this.keySettingsForArtifactUserPrompt)
                        .params(Map.of("numero", numChoices)))
                .templateRenderer(StTemplateRenderer.builder().startDelimiterToken('{')
                        .endDelimiterToken('}')
                        .build())
                .call()
                .content();

        ...
```


Note

Implementazione servizio Gemini - III

```
    ...
        String generatedArtifact = this.chatClient.prompt()
                .options(ChatOptions.builder()
                .model("gemini-2.0-flash")
                .temperature(1.0)
                //.topP(1.0)
                //.topK(30)
                .maxTokens(1024)
                //.frequencyPenalty(0.1)
                //.presencePenalty(0.1)
                .build())
                .user(u -> u.text(this.generatedArtifactPrompt)
                        .params(Map.of("lista", listResponse,
                                "numero", numParagraphs,
                                "artefatto", artifactRequest.artifact().type(),
                                "genere", artifactRequest.artifact().genre(),
                                "formato", converter.getFormat())))
                .templateRenderer(StTemplateRenderer.builder().startDelimiterToken('{')
                        .endDelimiterToken('}')
                        .build())
                .call()
                .content();

                return converter.convert(generatedArtifact);
    }

}
```


Note

Implementazione controllore REST

```
package it.venis.ai.spring.demo.controllers;

import org.springframework.web.bind.annotation.RestController;
...
import it.venis.ai.spring.demo.services.GeminiFromClientService;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

@RestController
public class QuestionFromClientController {

    private final GeminiFromClientService geminiService;

    ...

    @PostMapping("/client/advice")
    public Answer getSuggestionForArtifact(@RequestBody ArtifactRequest artifactRequest) {

        return this.geminiService.getSuggestionForArtifact(artifactRequest);

    }

    @PostMapping("/client/generate")
    public Artifact getGeneratedArtifact(@RequestBody ArtifactRequest artifactRequest,
            @RequestParam(required = true, defaultValue = "3") Integer numChoices,
            @RequestParam(required = true, defaultValue = "1") Integer numParagraphs) {

        return this.geminiService.getGeneratedArtifact(artifactRequest, numChoices, numParagraphs);

    }

}
```


Note

- Sistema di generazione di caratterizzazioni personaggi per un nuovo artefatto (tipo e genere)
  - Personaggio con nome, cognome, professione e carattere
  - Numero di personaggi parametrizzabile da *request*
  - REST API `/client/generate/character` o per i più temerari, lavorare direttamente su `/client/generate`

Note

Codice
Branch di riferimento

https://github.com/simonescannapieco/spring-ai-base-dgroove-venis-code.git

Branch: 16-spring-ai-gemini-step-back-prompting

Note