M18b - Langchain Hub

Anwendung Generativer KI

Stand: 05.2025

1 | Einführung

Der LangChain Hub ist eine zentrale Plattform für die Verwaltung, Versionierung und gemeinsame Nutzung von Prompts in LangChain-Anwendungen. Diese Funktion ermöglicht es Entwicklern, ihre Prompts zu standardisieren, zu optimieren und effizient in Teams zusammenzuarbeiten.

2 | Hauptfunktionen

2.1 Zentrale Prompt-Verwaltung

Der LangChain Hub ermöglicht die zentrale Speicherung und Verwaltung von Prompts, was besonders in größeren Teams oder Projekten vorteilhaft ist, um Konsistenz zu gewährleisten.

```
from langchain_hub import pull

# Einen bestimmten Prompt aus dem Hub abrufen
qa_prompt = pull("yourusername/qa_prompt")
```

2.2 Prompt-Versionierung

Ähnlich wie bei Git können Prompts versioniert werden, um Änderungen nachzuverfolgen und bei Bedarf zu früheren Versionen zurückzukehren.

```
# Eine bestimmte Version eines Prompts abrufen
qa_prompt_v2 = pull("yourusername/qa_prompt:v2")
```

2.3 Prompt-Sharing und Kollaboration

Teams können Prompts einfach teilen und gemeinsam daran arbeiten, was die Zusammenarbeit und Wiederverwendung von bewährten Prompt-Mustern fördert.

```
from langchain_hub import push
from langchain.prompts import PromptTemplate

# Einen neuen Prompt erstellen
my_prompt = PromptTemplate.from_template(
    "Analysiere folgendes Dataset: {dataset_description}"
)

# Prompt zum Hub hochladen
push(my_prompt, "yourusername/data_analysis_prompt", new_version=True)
```

2.4 Prompt-Bibliotheken und Community

Der Hub bietet Zugang zu einer wachsenden Sammlung von Community-erstellten Prompts für verschiedene Anwendungsfälle, die als Ausgangspunkt oder Inspiration dienen können.

```
# Einen Community-Prompt für Zusammenfassungen verwenden
summarization_prompt = pull("community/summarization:latest")
```

3 | Integration in LangChain-Anwendungen

Die Hub-Prompts lassen sich nahtlos in bestehende LangChain-Anwendungen integrieren:

```
from langchain_hub import pull
from langchain.chains import LLMChain
from langchain_openai import ChatOpenAI

# Prompt aus dem Hub abrufen
classification_prompt = pull("myteam/sentiment_classification:v3")

# LLM definieren
llm = ChatOpenAI(model="gpt-4")

# Chain erstellen und ausführen
chain = LLMChain(llm=llm, prompt=classification_prompt)
result = chain.run(text="Das neue Produkt übertrifft alle meine
Erwartungen!")
```

4 | Best Practices

4.1 Standardisierte Benennung

Entwickeln Sie ein konsistentes Benennungsschema für Ihre Prompts, das den Anwendungsfall und die Version klar kommuniziert.

```
teamname/anwendungsfall_prompttyp:version
```

Beispiel: `dataengineers/customer_feedback_classification:v2`

4.2 Dokumentation von Prompts

Fügen Sie jedem Prompt Metadaten und Dokumentation hinzu, um seinen Zweck, die erwarteten Eingaben und das Ausgabeformat zu beschreiben.

```
from langchain.prompts import PromptTemplate
from langchain_hub import push

prompt = PromptTemplate(
    template="Klassifiziere das folgende Kundenfeedback: {feedback}",
    input_variables=["feedback"],
    metadata={
        "description": "Prompt zur Klassifizierung von Kundenfeedback in
Kategorien",
        "expected_output": "Eine der Kategorien: Positiv, Neutral, Negativ",
        "example_inputs": {"feedback": "Ihr Produkt funktioniert
großartig!"}
    }
)

push(prompt, "myteam/feedback_classification", new_version=True)
```

4.3 A/B-Tests und Prompt-Experimente

Nutzen Sie den Hub für A/B-Tests verschiedener Prompt-Formulierungen, um die effektivsten Prompts für Ihre Anwendungsfälle zu identifizieren.

```
# Variante A abrufen und testen
prompt_a = pull("myteam/product_summary:v1")
# Variante B abrufen und testen
prompt_b = pull("myteam/product_summary:experimental")

# Vergleich der Ergebnisse beider Prompts
```

5 | Fortgeschrittene Anwendungen

5.1 Prompt-Chaining mit dem Hub

Komplexe Reasoning-Ketten können durch die Kombination spezialisierter Prompts aus dem Hub erstellt werden.

```
from langchain_hub import pull
from langchain.chains import SequentialChain
from langchain.chains import LLMChain
from langchain_openai import ChatOpenAI

llm = ChatOpenAI(model="gpt-4")

# Spezialisierte Prompts aus dem Hub abrufen
extraction_prompt = pull("myteam/data_extraction:v2")
analysis_prompt = pull("myteam/data_analysis:v1")
recommendation_prompt = pull("myteam/recommendations:v3")

# Chains erstellen
extraction_chain = LLMChain(llm=llm, prompt=extraction_prompt,
output_key="extracted_data")
analysis_chain = LLMChain(llm=llm, prompt=analysis_prompt,
output_key="analysis_results")
recommendation_chain = LLMChain(llm=llm, prompt=recommendation_prompt,
output_key="recommendations")

# Sequentielle Chain erstellen
sequential_chain = SequentialChain(
    chains=[extraction_chain, analysis_chain, recommendation_chain],
    input_variables=["raw_data"],
    output_variables=["recommendations"],
    verbose=True
)

results = sequential_chain({"raw_data": customer_data})
```

5.2 Dynamische Prompt-Auswahl

Implementieren Sie eine dynamische Auswahl von Prompts basierend auf Eingabedaten oder Kontextinformationen.

```
from langchain_hub import pull

def select_prompt(input_data):
    if "technical" in input_data.lower():
        return pull("myteam/technical_support:latest")
```

```
    elif "billing" in input_data.lower():
        return pull("myteam/billing_support:latest")
    else:
        return pull("myteam/general_support:latest")

user_query = "Ich habe ein technisches Problem mit der API-Integration."
selected_prompt = select_prompt(user_query)
# Verwenden des ausgewählten Prompts für die Antwortgenerierung
```

6 | Hub-Prompts mit Reasoning-Modellen

Für komplexe Reasoning-Aufgaben können spezialisierte Prompts aus dem Hub besonders wertvoll sein:

```
from langchain_hub import pull
from langchain_openai import ChatOpenAI
from langchain.chains import LLMChain

# Reasoning-Prompt aus dem Hub abrufen
reasoning_prompt = pull("myteam/step_by_step_reasoning:v2")

# Modell mit erweitertem Reasoning verwenden
reasoning_model = ChatOpenAI(model="gpt-4", temperature=0.2)

# Chain erstellen
reasoning_chain = LLMChain(llm=reasoning_model, prompt=reasoning_prompt)

# Komplexes Problem lösen mit strukturiertem Reasoning
problem = """
Ein Datensatz enthält 5000 Kundenrezensionen. 20% der Rezensionen enthalten
technische Fragen,
35% beziehen sich auf Preisgestaltung, und der Rest betrifft Lieferprobleme.
Welcher ML-Ansatz wäre optimal für die Klassifizierung dieser Daten und
warum?
"""

analysis = reasoning_chain.run(problem=problem)
```

7 | Template Repositories

LangChain Hub bietet vorgefertigte Template Repositories, die schnell in eigenen Projekten eingesetzt werden können:

```
# Community-Template für Few-Shot-Prompting abrufen
few_shot_template = pull("community/few_shot_classification:latest")
```

```
# Anpassen mit eigenen Beispielen
custom_few_shot = few_shot_template.format(
    examples=[
        {"input": "Das System reagiert nicht.", "output": "Technisches
Problem"},
        {"input": "Die Kosten sind zu hoch.", "output": "Preisbeschwerde"},
        {"input": "Tolle Funktionen!", "output": "Positives Feedback"}
    ]
)
```

8 | Praktische Anwendungsfälle

8.1 Mehrsprachige Prompts

Verwalten Sie Prompts in verschiedenen Sprachen für internationale Anwendungen:

```
# Sprachspezifische Prompts abrufen
de_prompt = pull("myteam/customer_service:de")
en_prompt = pull("myteam/customer_service:en")
es_prompt = pull("myteam/customer_service:es")

# Dynamische Sprachauswahl
def get_language_prompt(language_code):
    return pull(f"myteam/customer_service:{language_code}")
```

8.2 Domänenspezifische Prompts

Erstellen Sie spezialisierte Sammlungen von Prompts für verschiedene Geschäftsbereiche:

```
# Prompts für verschiedene ML-Aufgaben
classification_prompt = pull("ml_team/text_classification:v2")
sentiment_prompt = pull("ml_team/sentiment_analysis:latest")
entity_extraction_prompt = pull("ml_team/entity_extraction:stable")
```

9 | Zukunft des Prompt-Engineerings

Der LangChain Hub entwickelt sich kontinuierlich weiter und bietet Entwicklern leistungsstarke Werkzeuge für das Prompt-Engineering:

- **Automatisierte Prompt-Optimierung**: Zukünftige Versionen werden voraussichtlich Tools für automatisierte A/B-Tests und Optimierung von Prompts enthalten.
- **Erweiterte Prompt-Templates**: Komplexere Template-Strukturen mit bedingter Logik und dynamischer Formatierung.
- **KI-gestützte Prompt-Vorschläge**: Intelligente Vorschläge zur Verbesserung von Prompts basierend auf Erfolgsmetriken.

Der LangChain Hub bietet eine robuste Infrastruktur für die Verwaltung und Optimierung von Prompts, die besonders bei der Zusammenarbeit in Teams und bei der Entwicklung komplexer KI-Anwendungen wertvoll ist. Durch die Standardisierung und Versionierung von Prompts können Entwickler konsistentere und effektivere Ergebnisse erzielen.