Thực hành Xây dựng AI Agent “Sales Manager” với Guardrails

Mục tiêu:

- Tạo một agent gửi email chào hàng tự động.
- Ngăn chặn đầu vào chứa thông tin nhạy cảm như tên người thật (input guardrail).
- Đảm bảo đầu ra không chứa cụm từ lỗi thời (output guardrail).

I. Cài đặt Môi trường

```
pip install langchain langchain-core openai guardrails-ai
```

II. Cấu hình Guardrails

a. Input Guardrail – Kiểm tra Tên Riêng

Tạo file `input_guardrail.xml`:

```
<guardrail>
  <input>
    <validation name="no_personal_name" on_fail="exception">
      <regex pattern="\b(Alice|Bob|Charlie|David)\b" match="false"/>
    </validation>
  </input>
</guardrail>
```

Regex ở đây sẽ từ chối đầu vào có chứa các tên người thường gặp.

b. Output Guardrail – Chặn Từ Lỗi Thời

Tạo file `output_guardrail.xml`:

```
<guardrail>
  <output>
    <validation name="no_copyright" on_fail="exception">
      <regex pattern="copyright 2023" match="false"/>
    </validation>
  </output>
</guardrail>
```

III. Code Agent “Sales Manager”

Tạo file `sales_manager.py`:

```
from guardrails import Guard
from langchain.chat_models import ChatOpenAI
from langchain.schema import HumanMessage

# Load Guardrails
input_guard = Guard.from_rail("input_guardrail.xml")
output_guard = Guard.from_rail("output_guardrail.xml")

# Model
llm = ChatOpenAI(model="gpt-3.5-turbo")

def run_sales_email(prompt):
    # Kiểm tra input
    input_guard.validate(prompt)

    # Sinh email
    messages = [HumanMessage(content=prompt)]
    response = llm(messages).content

    # Kiểm tra output
    output_guard.validate(response)

    return response

# TEST 1 – Gây lỗi (Tên người thật)
try:
    print("▶ Test 1 – Có tên thật")
    prompt = "Viết email chào hàng gửi CEO từ Alice"
```

```
    print(run_sales_email(prompt))
except Exception as e:
    print("🛡️ Guardrail kích hoạt:", e)

# TEST 2 – Thành công
try:
    print("\n▶ Test 2 – Không có PII")
    prompt = "Viết email chào hàng gửi CEO từ Trưởng phòng Kinh doanh"
    print(run_sales_email(prompt))
except Exception as e:
    print("🛡️ Guardrail kích hoạt:", e)
```

IV. Kết quả mong đợi

- **Test 1** sẽ **bị từ chối**, vì có tên thật “Alice” → input guardrail phát hiện.
- **Test 2** sẽ **thành công**, tạo ra email phù hợp → output guardrail không bị kích hoạt.

V. Mở rộng

- Structured output (đầu ra có cấu trúc) là cách giúp mô hình AI **trả về dữ liệu ở dạng JSON** thay vì chỉ là một đoạn văn bản thuần túy. Điều này cực kỳ hữu ích trong các ứng dụng thực tế vì:
  - Bạn dễ dàng **xử lý** dữ liệu bằng code (Python, JS, v.v.).
  - Có thể hiển thị nội dung lên giao diện một cách có tổ chức.
  - Dễ dàng lưu vào cơ sở dữ liệu.

Mục tiêu

Tạo một AI Agent dạng Sales Manager, khi được yêu cầu **viết email chào hàng**, sẽ **trả về dữ liệu JSON có cấu trúc** như sau:

```
{
  "subject": "Let's Talk About Increasing Your ROI",
  "body": "Dear CEO, I hope this message finds you well...",
  "signature": "Head of Business Development"
}
```

Cách làm: Dùng Pydantic + Output Parser

Langchain hỗ trợ xuất ra JSON theo schema định nghĩa bởi Pydantic.

Bước 1: Cài thư viện cần thiết

```
pip install langchain openai pydantic
```

Bước 2: Tạo Schema bằng Pydantic

```
from pydantic import BaseModel

class SalesEmail(BaseModel):
    subject: str
    body: str
    signature: str
```

Bước 3: Dùng `OutputFixingParser` để ép LLM xuất JSON

```
from langchain.chat_models import ChatOpenAI
from langchain.output_parsers import PydanticOutputParser
from langchain.prompts import PromptTemplate

llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)

# Khai báo parser
parser = PydanticOutputParser(pydantic_object=SalesEmail)

# Tạo prompt
prompt = PromptTemplate(
    template="""Bạn là một chuyên gia viết email chào hàng. Viết một email theo định dạng JSON sau:

{format_instructions}

Thông tin:
- Gửi cho: CEO
- Người gửi: Trưởng phòng Kinh doanh
- Sản phẩm: Giải pháp tăng hiệu quả bán hàng

Bắt đầu viết email.
""",
    input_variables=[],
    partial_variables={"format_instructions": parser.get_format_instructions()}
)

# Gọi LLM
```

```
formatted_prompt = prompt.format()
output = llm.predict(formatted_prompt)

# Parse kết quả
email = parser.parse(output)

# In ra
print(email.json(indent=2))
```

Kết quả mong đợi

```
{
  "subject": "Giải pháp giúp CEO tăng hiệu quả bán hàng",
  "body": "Kính gửi CEO, Chúng tôi xin giới thiệu...",
  "signature": "Trưởng phòng Kinh doanh"
}
```

Phiên bản #2
Được tạo 2 tháng 5 2025 03:48:05 bởi Đỗ Ngọc Tú
Được cập nhật 2 tháng 5 2025 07:45:21 bởi Đỗ Ngọc Tú