Constrained and Guided Generation Tutorial

Overview

This tutorial explores the concepts of constrained and guided generation in the context of large language models. We'll focus on techniques to set up constraints for model outputs and implement rule-based generation using OpenAI's GPT models and the LangChain library.

Motivation

While large language models are powerful tools for generating text, they sometimes produce outputs that are too open-ended or lack specific desired characteristics. Constrained and guided generation techniques allow us to exert more control over the model's outputs, making them more suitable for specific tasks or adhering to certain rules and formats.

Key Components

1. Setting up constraints for model outputs
2. Implementing rule-based generation
3. Using LangChain's PromptTemplate for structured prompts
4. Leveraging OpenAI's GPT models for text generation

Method Details

We'll use a combination of prompt engineering techniques and LangChain's utilities to implement constrained and guided generation:

1. We'll start by setting up the environment and importing necessary libraries.
2. We'll create structured prompts using LangChain's PromptTemplate to guide the model's output.
3. We'll implement constraints by specifying rules and formats in our prompts.
4. We'll use OpenAI's GPT model to generate text based on our constrained prompts.
5. We'll explore different techniques for rule-based generation, including output parsing and regex-based validation.

Conclusion

By the end of this tutorial, you'll have a solid understanding of how to implement constrained and guided generation techniques. These skills will enable you to create more controlled and specific outputs from large language models, making them more suitable for a wide range of applications where precise and rule-adherent text generation is required.

Setup

First, let's import the necessary libraries and set up our environment.

In [1]:

```
import os
import re

from langchain_openai import ChatOpenAI
from langchain.prompts import PromptTemplate
from langchain.output_parsers import RegexParser

from dotenv import load_dotenv
load_dotenv()

# Set up the OpenAI API key
os.environ["OPENAI_API_KEY"] = os.getenv('OPENAI_API_KEY')

# Initialize the language model
llm = ChatOpenAI(model="gpt-4o-mini")

# Function to display model outputs
def display_output(output):
    """Display the model's output in a formatted manner."""
    print("Model Output:")
    print("-" * 40)
    print(output)
    print("-" * 40)
    print()
```

Setting Up Constraints for Model Outputs

Let's start by creating a constrained prompt that generates a product description with specific requirements.

In [2]:

```
constrained_prompt = PromptTemplate(
    input_variables=["product", "target_audience", "tone", "word_limit"]
    template="""Create a product description for {product} targeted at {
    Use a {tone} tone and keep it under {word_limit} words.
    The description should include:
    1. A catchy headline
    2. Three key features
    3. A call to action

    Product Description:
    """
)
```

```
# Generate the constrained output
input_variables = {
    "product": "smart water bottle",
    "target_audience": "health-conscious millennials",
    "tone": "casual and friendly",
    "word_limit": "75"
}

chain = constrained_prompt | llm
output = chain.invoke(input_variables).content
display_output(output)
```

```
Model Output:
----------------------------------------
**Stay Hydrated, Stay Awesome!**

Meet your new hydration buddy! Our Smart Water Bottle tracks your water in
take, reminds you to sip throughout the day, and syncs with your favorite
fitness apps. Made from eco-friendly materials and designed for on-the-go
lifestyles, it's the perfect accessory for health-conscious millennials. R
eady to elevate your hydration game? Grab yours today and drink up the goo
d vibes!
----------------------------------------
```

Implementing Rule-Based Generation

Now, let's implement a rule-based generation system for creating structured job postings.

In [3]:

```
job_posting_prompt = PromptTemplate(
    input_variables=["job_title", "company", "location", "experience"],
    template="""Create a job posting for a {job_title} position at {comp
    The candidate should have {experience} years of experience.
    Follow these rules:
    1. Start with a brief company description (2 sentences)
    2. List 5 key responsibilities, each starting with an action verb
    3. List 5 required qualifications, each in a single sentence
    4. End with a standardized equal opportunity statement

    Format the output as follows:
    COMPANY: [Company Description]

    RESPONSIBILITIES:
    - [Responsibility 1]
    - [Responsibility 2]
    - [Responsibility 3]
    - [Responsibility 4]
    - [Responsibility 5]

    QUALIFICATIONS:
    - [Qualification 1]
    - [Qualification 2]
    - [Qualification 3]
    - [Qualification 4]
    - [Qualification 5]
```

```
    EEO: [Equal Opportunity Statement]
    """
)

# Generate the rule-based output
input_variables = {
    "job_title": "Senior Software Engineer",
    "company": "TechInnovate Solutions",
    "location": "San Francisco, CA",
    "experience": "5+"
}

chain = job_posting_prompt | llm
output = chain.invoke(input_variables).content
display_output(output)
```

```
Model Output:
----------------------------------------
COMPANY: TechInnovate Solutions is a forward-thinking technology firm dedi
cated to developing cutting-edge software solutions that drive success for
businesses worldwide. Located in the heart of San Francisco, we pride ours
elves on fostering a collaborative and innovative work environment.

RESPONSIBILITIES:
- Design and implement robust software architectures to support scalable a
pplications.
- Lead cross-functional teams in the development and deployment of new fea
tures and enhancements.
- Collaborate with product managers to define and prioritize product requi
rements.
- Mentor junior engineers and provide guidance on best coding practices an
d methodologies.
- Conduct code reviews and ensure adherence to industry standards and qual
ity assurance processes.

QUALIFICATIONS:
- A minimum of 5 years of professional software engineering experience is
required.
- Proficiency in programming languages such as Java, Python, or JavaScript
is essential.
- Strong understanding of software development methodologies, including Ag
ile and DevOps practices.
- Experience with cloud platforms such as AWS, Azure, or Google Cloud is p
referred.
- Excellent problem-solving skills and the ability to work effectively in
a team-oriented environment are necessary.

EEO: TechInnovate Solutions is an equal opportunity employer. We celebrate
diversity and are committed to creating an inclusive environment for all e
mployees.
----------------------------------------
```

Using Regex Parser for Structured Output

Let's use a regex parser to ensure our output adheres to a specific structure.

In [7]:

```
# Define a regex parser for structured output
```

```
regex_parser = RegexParser(
    regex=r"COMPANY:\s*([\s\S]*?)\n\s*RESPONSIBILITIES:\s*([\s\S]*?)\n\s
    output_keys=["company_description", "responsibilities", "qualificati
)
# This regex pattern captures the company description, responsibilities,

# Create a new prompt template that includes the parser instructions
parsed_job_posting_prompt = PromptTemplate(
    input_variables=["job_title", "company", "location", "experience"],
    template="""Create a job posting for a {job_title} position at {comp
    The candidate should have {experience} years of experience.
    Follow these rules:
    1. Start with a brief company description (2 sentences)
    2. List 5 key responsibilities, each starting with an action verb
    3. List 5 required qualifications, each in a single sentence
    4. End with a standardized equal opportunity statement

    Format the output EXACTLY as follows:
    COMPANY: [Company Description]

    RESPONSIBILITIES:
    - [Responsibility 1]
    - [Responsibility 2]
    - [Responsibility 3]
    - [Responsibility 4]
    - [Responsibility 5]

    QUALIFICATIONS:
    - [Qualification 1]
    - [Qualification 2]
    - [Qualification 3]
    - [Qualification 4]
    - [Qualification 5]

    EEO: [Equal Opportunity Statement]
    """
)

def clean_output(output):
    for key, value in output.items():
        if isinstance(value, str):
            # Remove leading/trailing whitespace and normalize newlines
            output[key] = re.sub(r'\n\s*', '\n', value.strip())
    return output

# Generate the parsed output
chain = parsed_job_posting_prompt | llm
raw_output = chain.invoke(input_variables).content

# Parse and clean the output
parsed_output = regex_parser.parse(raw_output)
cleaned_output = clean_output(parsed_output)

# Display the parsed output
print("Parsed Output:")
for key, value in cleaned_output.items():
    print(f"{key.upper()}:")
    print(value)
    print()
```

```
Parsed Output:
COMPANY_DESCRIPTION:
TechInnovate Solutions is a leading technology firm based in San Francisc
o, CA, dedicated to creating cutting-edge software solutions that empower
businesses to thrive in the digital age. Our team of innovative thinkers a
nd problem solvers is committed to pushing the boundaries of technology to
deliver exceptional products and services.

RESPONSIBILITIES:
- Design and develop scalable software applications that meet the needs of
our clients.
- Collaborate with cross-functional teams to define, design, and implement
new features.
- Mentor junior engineers, providing guidance and support for their profes
sional growth.
- Troubleshoot and resolve software defects and performance issues in a ti
mely manner.
- Stay updated with emerging technologies and industry trends to ensure be
st practices.

QUALIFICATIONS:
- A minimum of 5 years of experience in software development, with a stron
g focus on full-stack technologies.
- Proficiency in programming languages such as Java, Python, or JavaScrip
t, along with relevant frameworks.
- Experience with cloud platforms such as AWS, Azure, or Google Cloud.
- Strong understanding of software development life cycle (SDLC) and agile
methodologies.
- Excellent problem-solving skills and ability to work in a fast-paced env
ironment.

EEO_STATEMENT:
TechInnovate Solutions is an equal opportunity employer. We celebrate dive
rsity and are committed to creating an inclusive environment for all emplo
yees.
```

Implementing Additional Constraints

Let's create a more complex constrained generation task: generating a product review with specific criteria.

In [8]:

```
review_prompt = PromptTemplate(
    input_variables=["product", "rating", "pros", "cons", "word_limit"],
    template="""Write a product review for {product} with the following
    1. The review should have a {rating}-star rating (out of 5)
    2. Include exactly {pros} pros and {cons} cons
    3. Use between 2 and 3 sentences for each pro and con
    4. The entire review should be under {word_limit} words
    5. End with a one-sentence recommendation

    Format the review as follows:
    Rating: [X] out of 5 stars

    Pros:
    1. [Pro 1]
    2. [Pro 2]
    ...
```

```
    Cons:
    1. [Con 1]
    2. [Con 2]
    ...

    Recommendation: [One-sentence recommendation]
    """
)

# Generate the constrained review
input_variables = {
    "product": "Smartphone X",
    "rating": "4",
    "pros": "3",
    "cons": "2",
    "word_limit": "200"
}

chain = review_prompt | llm
output = chain.invoke(input_variables).content
display_output(output)
```

```
Model Output:
----------------------------------------
Rating: 4 out of 5 stars

Pros:
1. The camera quality on Smartphone X is outstanding, capturing vibrant co
lors and sharp details even in low light. The multiple lens options provid
e versatility for different photography styles, making it a great choice f
or both casual users and photography enthusiasts.
2. Battery life is impressive, lasting a full day with heavy usage. Quick
charging capabilities ensure that you can get back to using your phone in
no time, which is a huge plus for those on the go.
3. The sleek design and lightweight build make Smartphone X comfortable to
hold and use throughout the day. Its premium feel and modern aesthetics al
so make it visually appealing.

Cons:
1. Although the performance is generally smooth, there can be occasional l
ag when multitasking with resource-heavy applications. This might be a dra
wback for users who rely heavily on their devices for productivity.
2. The lack of expandable storage is a limitation for those who need extra
space for apps, photos, and videos. Users may find themselves needing to m
anage their storage more frequently as a result.

Recommendation: Overall, Smartphone X is a fantastic choice for anyone see
king a powerful and stylish device.
----------------------------------------
```