STRUCTURED TEXT PARSING PROOF OF CONCEPT DOCUMENTATION

AUTHOR: ASHIK ROSHAN I

INTRODUCTION:

This Proof of Concept demonstrates the implementation of a structured text parser using the LangChain library. The parser is designed to extract specific information, such as names and ages, from a given text.

CODE EXPLANATION:

1. Importing Libraries:

```
In [27]: from langchain.output_parsers import ResponseSchema
         from langchain.output_parsers import StructuredOutputParser
```

Here, we import necessary modules from the LangChain library for structured output parsing.

2. Defining Response Schemas:

```
name_schema = ResponseSchema(name="Name",
                             description="Get the Names mentioned in the text.")
age_schema = ResponseSchema(name="Name",
                             description="Get the ages mentioned in the text.")
```

Two Response Schemas, **name_schema** and **age_schema**, are defined to specify the information to be extracted (Names and Ages) and provide brief descriptions.

3. Creating Output Parser:

```
response_schemas = [name_schema, age_schema]
output_parser = StructuredOutputParser.from_response_schemas(response_schemas)
```

An Output Parser is created using the specified Response Schemas, enabling structured information extraction.

4. Fetching Format Instructions:

```
format_instructions = output_parser.get_format_instructions()
```

Format instructions are obtained from the output parser for further use.

5. **Defining Sample Text:**

```
text = """\
    There are two friends named Ashik and Roshan with ages of 23 and 21 living in Madurai.
"""
```

A sample text containing information about names and ages is provided for testing.

6. **Creating Prompt Template:**

```
review_template = """\
For the following text, extract the following information:

Name: Get the Names mentioned in the text.
Age: Get the ages mentioned in the text.
text: {text}

{format_instructions}
"""
```

A prompt template is defined to guide the information extraction, including placeholders for the text and format instructions.

7. **Generating Chat Prompt:**

```
prompt = ChatPromptTemplate.from_template(template=review_template)
messages = prompt.format_messages(text=text, format_instructions=format_instructions)
```

A chat prompt is generated using the template and formatted with the sample text and format instructions.

8. **Interacting with the Chat Model:**

```
response = chat(messages)
```

The formatted prompt is sent to the chat model for processing.

9. **Displaying Results:**

```
print(response.content)
```

The model's response, containing the extracted information, is printed.

10. **Extracted Information:**

The parsed information, including names and ages, is extracted and displayed based on the provided sample text.

```json
{
        "Name": ["Ashik", "Roshan"],
        "Age": [23, 21]
}
```

Conclusion:

This Proof of Concept demonstrates the successful implementation of a structured text parser using LangChain, showcasing the extraction of specific information (names and ages) from a given text.