Project Documentation Template

Project Title: Text Summarizer Using Asyncflows and Gradio

Table of Contents

Introduction

The project aims to provide a user-friendly interface for summarizing text inputs using advanced natural language processing techniques. It utilizes Asyncflows for managing asynchronous actions and Gradio for creating an interactive UI, enabling easy text summarization for various applications.

Problem Statement

Manual text summarization is time-consuming and prone to errors, especially with large volumes of text. Automated solutions are needed to efficiently condense textual content while preserving key information and context.

Solution Overview

Our solution leverages Asyncflows for handling asynchronous tasks within the summarization pipeline. It integrates Gradio to offer an intuitive web-based interface where users can input text and receive concise summaries promptly.

Target Audience

- **Researchers:** To quickly extract key findings from academic papers.
- **Students:** To summarize study materials and lecture notes.
- **Professionals:** To generate briefs and reports from extensive documents.
- **Developers:** To integrate automated text summarization into other applications.

Project Setup

Prerequisites

- Python 3.8+
- Asyncflows
- Gradio
- NLTK (Natural Language Toolkit)

- Other dependencies as specified in `requirements.txt`

Installation

Provide step-by-step instructions for setting up the development environment and installing necessary dependencies.

Example:

```
# Clone the repository
git clone https://github.com/Kirai-Kevin/textsum.git

# Navigate to the project directory
cd textsum

# Install dependencies
pip install -r requirements.txt
```

Running the Project

Explain how to run the pr

oject.

Example:

```
# Start the application
python main.py
```

Flow Design

Flow Diagram

Include your flow diagram here depicting the steps involved in the text summarization process using Asyncflows.

Flow Description

1. **Input Handling:** Accepts input from users via a Gradio interface or text file upload.
2. **Text Processing:** Cleans and preprocesses the input text to remove noise and irrelevant content.
3. **Asynchronous Summarization:** Utilizes Asyncflows to asynchronously summarize the preprocessed text using NLP models.
4. **Output Generation:** Generates a summarized version of the input text suitable for quick review and understanding.

Custom Action

Description

The custom action focuses on asynchronously summarizing text inputs using Asyncflows for efficient execution.

Code

```
import asyncio

from transformers import pipeline



class BaseAsyncAction:

    async def execute(self, data):

        raise NotImplementedError("Subclasses should implement this
method")



class SummarizeTextAction(BaseAsyncAction):

    def __init__(self):

        self.summarizer = pipeline("summarization")



    async def execute(self, text: str) -> str:
```

```
        return await self.summarize_text(text)  # Ensure 'await' is used
correctly



    async def summarize_text(self, text: str) -> str:

        """

        Asynchronously generates a summary for the provided text.



        Parameters:

        - text (str): The text to summarize.



        Returns:

        - str: The summarized text.

        """

        await asyncio.sleep(1)  # Simulate processing time

        summary_list = self.summarizer(text, max_length=150,
min_length=40, do_sample=False)

        return summary_list[0]['summary_text']
```

Integration

Integrate the custom action into the Asyncflows-based flow for text summarization:

```
import asyncio
```

```
class Flow:

    def __init__(self):

        self.actions = {}



    def add_action(self, name, action):

        self.actions[name] = action



    async def execute_action(self, name, data):

        action = self.actions.get(name)

        if action:

            return await action.execute(data)  # Ensure 'await' is used
correctly

        else:

            raise ValueError(f"Action '{name}' not found.")
```

Reusability

The `summarize_text_async` action can be easily reused in other projects or workflows that require text summarization capabilities. It abstracts the NLP model interaction and summarization logic, making it modular and adaptable.

User Interface

UI Design

Describe the design and layout of the user interface. Include screenshots if available.

Gradio Implementation

Explain how Gradio was used to create the user interface for interacting with the project:

```
def gr_interface(input_text, input_pdf):
    summary, pdf_bytes = asyncio.run(summarize_input(input_text,
input_pdf))
    return summary

# Route for downloading PDF
@app.route('/download-pdf')
def download_pdf():
    _, pdf_bytes = asyncio.run(summarize_input("", None))
    return send_file(pdf_bytes, as_attachment=True,
attachment_filename="summary.pdf")

# Gradio interface setup
iface = gr.Interface(
    fn=gr_interface,
    inputs=[
        gr.Textbox(lines=10, placeholder="Paste text here..."),
        gr.File(label="Upload PDF")
    ],
    outputs="text",
    title="Text Summarizer",
    description="Upload a PDF file or paste text to get a summary.
Download the summary as a PDF <a href='/download-pdf'
target='_blank'>here</a>."
)

# Launch the interface
iface.launch(inbrowser=True, share=True, debug=True)
```

How to Use

1. **Start the Application:** Run `python main.py` to launch the application.
2. **Input Data:** Enter or paste text into the provided textbox.
3. **Generate Output:** Click on the "Summarize" button to initiate the text summarization process.
4. **View and Download Output:** The summarized text will be displayed. Optionally, download the output for further use.

Use Cases

1. **Academic Research:** Extract key findings and conclusions from research papers.
2. **Educational Purposes:** Summarize study materials and lecture notes for better understanding.
3. **Professional Reports:** Quickly generate executive summaries from lengthy documents.

Demo Video

Include a link to your demo video. The video should demonstrate the key features of your project and explain how it works.

Demo Video Link

Conclusion

The Text Summarizer project demonstrates effective use of Asyncflows and Gradio to automate text summarization tasks, providing users with an efficient solution for information extraction from textual content.

Future Work

- **Enhanced NLP Models:** Integrate advanced NLP models for improved summarization accuracy and context understanding.
- **Multilingual Support:** Extend text summarization capabilities to handle documents in multiple languages.

- **Customization Options:** Provide more options for user-defined summarization preferences and settings.
- **Integration with External APIs:** Incorporate APIs for additional data enrichment and specialized summarization tasks.

References

Asyncflows Documentation: https://asyncflow.readthedocs.io/en/latest/

Gradio Documentation: https://www.gradio.app/docs

NLTK Documentation: https://www.nltk.org/

Transformers Documentation: https://huggingface.co/docs/transformers/en/index

---

Use this template to document your project thoroughly. Ensure that each section is detailed and provides a clear understanding of your work.