Documentation: Surya's Langchain: Summarize Text from YouTube and Website

Overview

This is a Streamlit-based application that utilizes LangChain and Groq's LLM to summarize content from YouTube videos and web pages. Users can input a YouTube or website URL, and the app fetches the content, processes it, and provides a concise summary.

Features

- **User-Friendly Interface**: Streamlit-based UI for easy interaction.
- **Groq API Integration**: Uses Groq's gemma2-9b-it model for text summarization.
- **YouTube Video Summarization**: Extracts transcripts from YouTube videos and summarizes them.
- **Website Content Summarization**: Retrieves and summarizes text content from web pages.
- **Error Handling**: Provides informative error messages for invalid URLs or missing API keys.

Workflow

1. **User Input**:
    - The user enters a YouTube or website URL.
    - A valid Groq API key must be provided.
2. **Validation**:
    - The app checks if the API key is provided.
    - It validates the URL to ensure it is correctly formatted.
    - If the URL is invalid, an error message is displayed.
3. **Content Extraction**:
    - If the URL is from YouTube, the app extracts the video transcript.
    - If the URL is a website, the app fetches the page content.
    - If content retrieval fails, appropriate error messages are displayed.

4. **Summarization**:
    - The extracted content is passed through a LangChain summarization pipeline.
    - A structured prompt is used to generate a 300-word summary.
5. **Output Display**:
    - The generated summary is displayed in the Streamlit interface.
    - Success and error messages help guide the user through the process.

Dependencies

- **Streamlit**: For building the web-based UI.
- **LangChain**: For prompt handling and LLM interaction.
- **LangChain-Groq**: To use Groq's LLM for text summarization.
- **LangChain-Community**: For document loaders to extract content from YouTube and websites.
- **Validators**: For checking the validity of user-inputted URLs.
- **OpenAI-Compatible API**: Groq API key is required to process the summarization.

Error Handling

- **Missing API Key**: The app stops execution and prompts the user to enter a valid key.
- **Invalid URL**: Displays an error if the provided URL is not a valid YouTube or website link.
- **Content Extraction Failures**: Specific error messages guide the user when the content cannot be fetched.
- **LLM Processing Errors**: If any issue occurs during summarization, the app notifies the user.

Use Cases

- **Quick Summaries**: Get a concise overview of long YouTube videos or web articles.
- **Research Assistance**: Summarize articles to extract key insights.
- **Time-Saving**: Helps users consume large amounts of information efficiently.

Future Improvements

- **Multi-Language Support**: Enable summarization for content in different languages.
- **Custom Summary Length**: Allow users to define the length of the summary.
- **Advanced NLP Models**: Experiment with different LLMs for better summarization quality.
- **Multiple URL Inputs**: Enable batch processing for multiple URLs.

Code:

```
import validators
import streamlit as st
from langchain.prompts import PromptTemplate
from langchain_groq import ChatGroq
from langchain.chains.summarize import load_summarize_chain
from langchain_community.document_loaders import YoutubeLoader,
UnstructuredURLLoader

# Creating Streamlit app
st.set_page_config(page_title="Surya's Langchain: Summarize Text from YT and Website")
st.title("Surya's Langchain: Summarize Text from YT and Website")
st.subheader("It Summarizes your topic")

# Sidebar - Groq API Key
with st.sidebar:
  groq_api_key = st.text_input("Groq API Key", value="", type="password")

# Input URL
generic_url = st.text_input("Enter URL Here")

# Validate API Key
if not groq_api_key.strip():
  st.error("Please provide a Groq API Key.")
```

```
    st.stop()

# Initialize LLM
llm = ChatGroq(model="gemma2-9b-it", groq_api_key=groq_api_key)

# Prompt Template
prompt_template = """
Provide a summary of the following content in 300 words:
Content: {text}
"""

prompt = PromptTemplate(template=prompt_template, input_variables=["text"])

# Summarization Button
if st.button("Summarize the Content"):
    if not generic_url.strip():
        st.error("Please provide the URL.")
    elif not validators.url(generic_url):
        st.error("Invalid URL! It must be a YouTube or website URL.")
    else:
        try:
            with st.spinner("Fetching and summarizing content..."):
                # Load Content (YouTube or Website)
                if "youtube.com" in generic_url or "youtu.be" in generic_url:
                    try:
                        loader = YoutubeLoader.from_youtube_url(
                            generic_url, add_video_info=False
                        )
                        data = loader.load()
                    except Exception as yt_error:
```

```
                st.error(f"Failed to fetch YouTube transcript: {yt_error}")
                data = []
        else:
            try:
                loader = UnstructuredURLLoader(urls=[generic_url])
                data = loader.load()
            except Exception as web_error:
                st.error(f"Failed to fetch website content: {web_error}")
                data = []

        # Ensure valid content is extracted
        if not data or not isinstance(data, list):
            st.error("Failed to retrieve content. Please check the URL and try again.")
        else:
            # Summarization Chain (Passing Correct Input)
            chain = load_summarize_chain(llm, chain_type="stuff", prompt=prompt)
            output_summary = chain.run({"input_documents": data})

            st.success("Summary Generated Successfully!")
            st.write(output_summary)

except Exception as e:
    st.error(f"An error occurred: {e}")
```