**Documentation: NLP Techniques and Implementation**

**Overview**

Code snippet creates a mental health chat bot using NLP (Natural Language Processing) techniques. The bot is designed to provide relaxation tips and stress relief techniques based on user input about their mood or stress level. Here’s a breakdown of the key components and NLP techniques used in your implementation:

**1. LangChain and NLP Components**

**LangChain** is a framework designed to facilitate the creation of NLP applications by chaining together various components like prompt templates, LLMs (Large Language Models), and output parsers. Here’s how each part of the code fits into LangChain:

1. **ChatPromptTemplate**: This defines how the input and output are structured. It uses a conversational template to guide the LLM in generating responses. In your case, it specifies the role of the assistant and formats the user’s input.
2. **Ollama (LLM)**: This represents the large language model that will process the input based on the prompt. It generates responses by understanding and responding to the user's mood or stress level.
3. **StrOutputParser**: This component is used to parse the output from the LLM. In this case, it handles the string responses to ensure they are formatted correctly for presentation.

**2. Detailed Code Explanation**

**2.1. Importing Libraries**

from langchain\_core.prompts import ChatPromptTemplate

from langchain\_core.output\_parsers import StrOutputParser

from langchain\_community.llms import Ollama

import streamlit as st

* ChatPromptTemplate helps structure the interaction with the LLM.
* StrOutputParser is used to handle and format the string output from the LLM.
* Ollama is an example of a large language model (LLM) used for generating responses.
* streamlit is used to create the web interface for user interaction.

**2.2. Streamlit Setup**

st.title("Mental Health Chat Bot")

input\_txt = st.text\_input("Please enter your mood or stress level...")

* st.title() sets the title of the web application.
* st.text\_input() provides a text box for users to input their mood or stress level.

**2.3. Defining the Prompt Template**

prompt = ChatPromptTemplate.from\_messages(

[

("system", "You are a mental health assistant named SwethaSangavi. Your task is to provide specific relaxation tips and stress relief techniques based on the user's mood or stress levels. Focus on actionable strategies to alleviate stress and anxiety."),

("user", "User input: {query}")

]

)

* This defines how the conversation should be structured.
* The "system" message sets the context for the LLM, specifying that it should provide stress relief tips.
* The "user" message includes a placeholder ({query}) for the user’s input.

**2.4. Initializing the LLM**

llm = Ollama(model="llama3")

* Initializes the LLM with the model named "llama3". This model will be used to generate responses based on the defined prompt.

**2.5. Creating the Chain**

chain = prompt | llm | output\_parser

* Creates a pipeline where the prompt template is applied to the input, the LLM generates a response, and the output parser formats the response.

**2.6. Processing User Input**

if input\_txt:

result = chain.invoke({"query": input\_txt})

st.write(result)

* Checks if the user has provided input.
* Uses the defined chain to process the input and generate a response.
* Displays the result in the Streamlit interface.

**3. NLP Techniques Used**

1. **Prompt Engineering**: Crafting specific prompts to guide the LLM in generating relevant and useful responses. In your case, the prompt template directs the LLM to focus on stress relief techniques.
2. **Conversational AI**: Using the LLM to simulate a conversation with the user, providing responses that are contextually appropriate based on the user’s mood or stress level.
3. **Output Parsing**: Handling and formatting the LLM’s output to ensure it is presented clearly and effectively to the user.

**Conclusion**

Your code leverages various NLP techniques and tools to build a mental health chat bot. By using LangChain components, you can create a structured pipeline that processes user inputs and generates helpful responses. This approach allows for scalable and flexible NLP applications, such as mental health support bots, by combining prompt engineering, conversational AI, and output parsing.