அத்தியாயம் 9: 'வளரி' – நமது தமிழ் AI உதவியாளரின் உருவாக்கம்

நமது செயற்கை நுண்ணறிவுப் பயணத்தில், நாம் ஏற்கெனவே இரண்டு முறை படைப்பாளிகளாக மாறியுள்ளோம். முதலில், விதிகளின் அடிப்படையில் ஒரு எளிய சாட்பாட் என்ற மரப் பொம்மையைச் செதுக்கினோம். அடுத்து, டீப் லேர்னிங் என்ற ஆழமான அறிவைக் கொண்டு, சங்கத் தமிழ் பாணியில் கவிதை பாடும் ஒரு AI புலவன் என்ற சிற்பத்தை வடித்தோம்.

இப்போது, நமது பயணத்தின் மூன்றாவது, மிக சக்திவாய்ந்த கட்டத்திற்கு வந்துள்ளோம்.

இந்த முறை, நாம் புதிதாக ஒரு சிற்பத்தைச் செதுக்கப் போவதில்லை. மாறாக, உலகின் தலைசிறந்த சிற்பி (OpenAI) உருவாக்கிய, எல்லையற்ற ஆற்றல் கொண்ட ஒரு பிரம்மாண்டமான டிஜிட்டல் மூளையை, எப்படி நமது தேவைக்கேற்பக் கையாள்வது, அதற்கு எப்படி ஒரு முகத்தைக் (Streamlit) கொடுத்து, நம்முடன் உரையாட வைப்பது என்பதைக் கற்கப் போகிறோம்.

இந்த அத்தியாயத்தில், நாம் '**வளரி**' என்ற நமது சொந்த தமிழ் AI உதவியாளரை உருவாக்குவோம். வாருங்கள், ஒரு மாபெரும் சக்தியை இயக்கும் கலையைக் கற்போம்.

9.1. தேவையான நூலகங்கள் (Libraries)

முதலில், நமக்குத் தேவையான நூலகங்களை இம்போர்ட் செய்ய வேண்டும். இந்த திட்டத்தில் பயன்படுத்தப்படும் நூலகங்கள்:

- **Streamlit**: இது ஒரு பைதான் நூலகம், இது டேட்டா அறிவியல் மற்றும் மெஷின் லேர்னிங் பயன்பாடுகளை எளிதாக உருவாக்க உதவுகிறது.
- **OpenAI**: இது OpenAI API ஐப் பயன்படுத்தி, GPT மாடல்களுடன் தொடர்பு கொள்ள உதவுகிறது.
- **os**: இது ஆப்பரேட்டிங் சிஸ்டம் லெவல் செயல்பாடுகளை நிர்வகிக்க உதவுகிறது.

```
import streamlit as st
from openai import OpenAI
import os
```

9.2. OpenAI API கீயை அமைத்தல்

OpenAI API ஐப் பயன்படுத்த, நமக்கு ஒரு API கீ தேவை. இந்த கீயை `os.environ` மூலம் அமைக்கலாம். இது பாதுகாப்பான வழியில் API கீயை சேமிக்க உதவுகிறது.

```
os.environ["OPENAI_API_KEY"] = "your-openai-api-key"
```

குறிப்பு: `"your-openai-api-key"` என்பதற்கு பதிலாக உங்கள் OpenAI API கீயை உள்ளிடவும்.

9.3. OpenAI மாடலைப் பயன்படுத்தி பதிலளித்தல்

இந்த பகுதியில், `continue_conversation` செயல்பாட்டைப் பயன்படுத்தி, OpenAI API ஐப் பயனரின் கேள்விகளுக்கு எவ்வாறு பதிலளிக்கிறது மற்றும் அதன் அளவுருக்கள் (parameters) எவ்வாறு செயல்படுகின்றன என்பதைப் படிப்படியாகப் புரிந்துகொள்வோம்.

செயல்பாட்டின் அமைப்பு

```
def continue_conversation(messages, temperature=0):
    client = OpenAI()
    response = client.chat.completions.create(
        model="gpt-4",  # GPT மாடல் பெயர்
        messages=messages,  # உரையாடல் வரலாறு
        temperature=temperature,  # பதிலின் படைப்பாற்றல் நிலை
    )
    return response.choices[0].message.content
```

இந்த செயல்பாடு இரண்டு அளவுருக்களை எடுக்கிறது:

1. **messages**: இது உரையாடல் வரலாறு. இது ஒரு பட்டியல் (list) ஆகும், இதில் பயனர் மற்றும் உதவியாளரின் செய்திகள் சேமிக்கப்படும்.
2. **temperature**: இது பதிலின் படைப்பாற்றல் நிலை. இது ஒரு எண் மதிப்பு (0 முதல் 1 வரை).

OpenAI கிளையண்டை உருவாக்குதல்

```
client = OpenAI()
```

இந்த வரி OpenAI API உடன் தொடர்பு கொள்ள ஒரு கிளையண்டை உருவாக்குகிறது. இந்த கிளையண்ட் மூலம், GPT மாடல்களுடன் தொடர்பு கொள்ளலாம்.

`client.chat.completions.create()` முறை

இந்த முறை OpenAI API க்கு கேள்வியை அனுப்பி, பதிலைப் பெற உதவுகிறது. இதற்கு மூன்று முக்கிய அளவுருக்கள் உள்ளன:

அ) `model="gpt-4"`

- **பயன்பாடு**: இது GPT மாடலின் பெயரைக் குறிக்கிறது.
- **விளக்கம்**: இங்கு `"gpt-4"` மாடலைப் பயன்படுத்துகிறோம். இதை `"gpt-3.5-turbo"` போன்ற வேறு மாடல்களாக மாற்றலாம்.
- **எடுத்துக்காட்டு**: `model="gpt-3.5-turbo"` என்று மாற்றினால், GPT-3.5 மாடல் பயன்படுத்தப்படும்.

ஆ) `messages=messages`

- **பயன்பாடு**: இது உரையாடல் வரலாற்றைக் குறிக்கிறது.
- **விளக்கம்**: `messages` என்பது ஒரு பட்டியல் (list), இதில் ஒவ்வொரு செய்தியும் ஒரு dictionary ஆகும். ஒவ்வொரு dictionary யும் இரண்டு key-களைக் கொண்டிருக்கும்:
  - **`role`**: செய்தியை அனுப்பியவர் யார் என்பதைக் குறிக்கிறது. இது `"system"`, `"user"`, அல்லது `"assistant"` ஆக இருக்கலாம்.
  - **`content`**: செய்தியின் உள்ளடக்கம்.
- **எடுத்துக்காட்டு**:

```
messages = [
    {"role": "system", "content": "நீங்கள் ஒரு தமிழ் உதவியாளர்."},
    {"role": "user", "content": "தமிழ் எப்படி கற்கலாம்?"}
]
```

இ) `temperature=temperature`

Language Generation மாடல்களில், **Temperature** என்பது ஒரு முக்கியமான அளவுரு. இது மாடல் எவ்வாறு சொற்களைத் தேர்ந்தெடுக்கிறது என்பதைக் கட்டுப்படுத்துகிறது. இதை எளிதாகப் புரிந்துகொள்ள ஒரு உதாரணத்துடன் விளக்குவோம்.

Temperature என்றால் என்ன?

Language Generation மாடல்கள், ஒரு வாக்கியத்தை உருவாக்கும்போது, ஒவ்வொரு சொல்லையும் தேர்ந்தெடுக்கும். இந்த தேர்வு, சொற்களின் "நிகழ்தகவு" (probability) அடிப்படையில் நடைபெறுகிறது. Temperature அளவுரு, இந்த நிகழ்தகவுகளை எவ்வாறு பயன்படுத்துவது என்பதைக் கட்டுப்படுத்துகிறது.

எப்படி செயல்படுகிறது?

மாடல் ஒரு வாக்கியத்தை உருவாக்கும்போது, ஒவ்வொரு சொல்லுக்கும் ஒரு நிகழ்தகவு மதிப்பைக் கணக்கிடுகிறது. உதாரணமாக, வாக்கியம்:
"தமிழ் மொழி..."
இதற்கு மாடல் பின்வரும் சொற்களைத் தேர்ந்தெடுக்கலாம்:

- **தொன்மையானது** — 50% நிகழ்தகவு
- **பழமையானது** — 30% நிகழ்தகவு
- **சிக்கலானது** — 15% நிகழ்தகவு
- **எளிமையானது** — 5% நிகழ்தகவு

இப்போது, Temperature மதிப்பு இந்த தேர்வை எவ்வாறு பாதிக்கிறது என்று பார்ப்போம்.

3. Temperature மதிப்புகள் மற்றும் அவற்றின் விளைவுகள்

அ) Temperature = 0

- **விளக்கம்**: Temperature 0 எனில், மாடல் எப்போதும் **அதிக நிகழ்தகவு உள்ள சொல்லைத் தேர்ந்தெடுக்கும்.**
- **எடுத்துக்காட்டு**:
    - சொல்: "**தொன்மையானது**" (50% நிகழ்தகவு)
    - வாக்கியம்: **"தமிழ் மொழி தொன்மையானது."**

பதில்கள் மிகவும் **துல்லியமாகவும், முன்னரே தீர்மானிக்கப்பட்டவையாகவும்** இருக்கும்.

ஆ) Temperature = 0.5

- **விளக்கம்**: Temperature 0.5 எனில், மாடல் **அதிக நிகழ்தகவு உள்ள சொற்களை அடிக்கடி தேர்ந்தெடுக்கும்**, ஆனால் சில சமயங்களில் குறைந்த நிகழ்தகவு உள்ள சொற்களையும் தேர்ந்தெடுக்கலாம்.
- **எடுத்துக்காட்டு**:
    - சொல்: "**பழமையானது**" (30% நிகழ்தகவு)
    - வாக்கியம்: "**தமிழ் மொழி பழமையானது.**"

பதில்கள் **படைப்பாற்றலுடனும், சிறிது மாறுபாடுடனும்** இருக்கும்.

இ) Temperature = 1

- **விளக்கம்**: Temperature 1 எனில், மாடல் **அனைத்து சொற்களையும் சமமான வாய்ப்புகளுடன் தேர்ந்தெடுக்கும்**. இது படைப்பாற்றலை அதிகரிக்கும், ஆனால் சில சமயங்களில் தவறான அல்லது பொருத்தமற்ற சொற்களையும் தேர்ந்தெடுக்கலாம்.
- **எடுத்துக்காட்டு**:
    - சொல்: "**எளிமையானது**" (5% நிகழ்தகவு)
    - வாக்கியம்: "**தமிழ் மொழி எளிமையானது.**"

Temperature அதிகரிக்கும் போது, பதில்கள் படைப்பாற்றலுடன் இருக்கும், ஆனால் துல்லியம் குறையலாம். Temperature அதிகமாக இருந்தால், மாடல் **தவறான அல்லது பொருத்தமற்ற தகவல்களை** (hallucinations) தரலாம்.

பதிலைத் திரும்பப் பெறுதல்

```
return response.choices[0].message.content
```

- **`response`**: OpenAI API இலிருந்து கிடைக்கும் பதில்.
- **`response.choices`**: இது பதில்களின் பட்டியல். ஒரே ஒரு பதில் இருப்பதால், `choices[0]` பயன்படுத்தப்படுகிறது.
- **`message.content`**: உதவியாளரின் பதில் உள்ளடக்கம்.

எடுத்துக்காட்டு

பயனர் கேள்வி: "தமிழ் எப்படி கற்கலாம்?"

```
messages = [
    {"role": "system", "content": "நீங்கள் ஒரு தமிழ் உதவியாளர்."},
    {"role": "user", "content": "தமிழ் எப்படி கற்கலாம்?"}
]

response = continue_conversation(messages, temperature=0.5)
print(response)
```

பதில்:

```
தமிழ் கற்க புத்தகங்கள் படியுங்கள், தமிழ் பாடல்கள் கேளுங்கள், மற்றும் தமிழ் பேசும்
நண்பர்களுடன் பேசுங்கள்.
```

9.4. Streamlit பயன்பாட்டை உருவாக்குதல்

Streamlit பயன்பாட்டை உருவாக்க, `main()` செயல்பாட்டை உருவாக்குவோம். இது பயனர் இன்டர்ஃபேஸைக் காட்டும்.

```
def main():
    st.title("வளரி - உங்கள் தனிப்பட்ட தமிழ் உதவியாளர்")
    st.write("உங்கள் கேள்விகளை தமிழில் கேளுங்கள், தமிழில் துல்லியமான
பதில்களைப் பெறுங்கள்!")
```

- **st.title()**: பயன்பாட்டின் தலைப்பைக் காட்டும்.
- **st.write()**: பயன்பாட்டின் விளக்கத்தைக் காட்டும்.

9.5. உரையாடல் வரலாற்றை நிர்வகித்தல்

Streamlit இல், `st.session_state` மூலம் உரையாடல் வரலாற்றை நிர்வகிக்கலாம். இது பயனர் மற்றும் உதவியாளரின் செய்திகளை சேமிக்க உதவுகிறது.

```
if "context" not in st.session_state:
    st.session_state.context = [{
        "role": "system",
```

```
        "content": """
        நீங்கள் எனது தனிப்பட்ட உதவியாளராக செயல்பட வேண்டும். உங்கள் பெயர் வளரி. நீங்கள் எப்போதும் நட்பாக, உதவியாகவும், சுவாரஸ்யமாகவும் இருக்க வேண்டும். நான் உங்களிடம் எதைக் கேட்டாலும், அதற்கு தெளிவான மற்றும் பயனுள்ள பதில்களை **தமிழில் மட்டும்** வழங்க வேண்டும்.

        நான் உணவு, தொழில்நுட்பம், புத்தகங்கள், திரைப்படங்கள் அல்லது வாழ்க்கை பற்றிய ஆலோசனை கேட்டால், அதற்கு ஏற்றவாறு பதிலளிக்க வேண்டும். நான் உணவு ஆர்டர் செய்ய விரும்பினால், அதை எளிதாகவும் வேடிக்கையாகவும் செய்ய உதவுங்கள். எனக்கு பரிந்துரைகள் கூறுங்கள், என் நாள் எப்படி இருந்தது என்று கேளுங்கள், மேலும் நான் விரும்புவதை சரியாகப் பெற உதவுங்கள்.

        நீங்கள் எப்போதும் இயல்பான மற்றும் நட்பு டோனில் பேச வேண்டும். நான் விவரங்களைக் கேட்டால், அவற்றை வழங்க தயங்க வேண்டாம். நான் கேட்கும் கேள்விக்கு பதில் தெரியாவிட்டால், உண்மையாக சொல்லுங்கள்—போலி பதில்கள் தர வேண்டாம்.

        **முக்கியமாக**, நான் வேறு மொழியில் பேசினாலும், நீங்கள் எப்போதும் தமிழில் மட்டுமே பதிலளிக்க வேண்டும். உதாரணமாக, நான் ஆங்கிலத்தில் கேள்வி கேட்டால், "நான் தமிழில் மட்டுமே பதிலளிக்க உருவாக்கப்பட்டுள்ளேன்!" என்று கூறி, தமிழில் பதிலை வழங்க வேண்டும்.

        நீங்கள் எப்போதும் எனது நம்பிக்கையான உதவியாளராக இருங்கள். நான் எதை வேண்டுமானாலும் கேட்கலாம், நீங்கள் அதை எளிதாகவும் வேடிக்கையாகவும் ஆக்குங்கள்!
        """
    }]
```

- **st.session_state.context**: இது உரையாடல் வரலாற்றை சேமிக்கும். முதல் செய்தி `system` ரோலில் உள்ளது, இது உதவியாளரின் நடத்தையை வரையறுக்கிறது.

9.6. பயனர் உள்ளீட்டைப் பெறுதல்

பயனர் தங்கள் கேள்வியை உள்ளிட, `st.text_input()` பயன்படுத்தப்படுகிறது.

```
user_input = st.text_input("உங்கள் கேள்வியை எழுத்து:")
```

9.7. பதிலைப் பெறுதல்

பயனர் "பதில் பெறுங்கள்" பொத்தானை அழுத்தினால் உரையாடலைப் புதுப்பித்து, OpenAI API ஐப் பயன்படுத்தி பதிலைப் பெறுவோம்.

```
if st.button("பதில் பெறுங்கள்"):
    with st.spinner("வளரி பதிலளிக்கிறது..."):
        st.session_state.context.append({"role": "user", "content": user_input})
        response = continue_conversation(st.session_state.context)
        st.session_state.context.append({"role": "assistant", "content": response})

        st.write("வளரி:")
        st.write(response)
```

- **st.button()**: பயனர் பொத்தானை அழுத்தினால் செயல்படும்.
- **st.spinner()**: பதில் பெறும் போது லோடிங் அனிமேஷன் காட்டும்.
- **st.write()**: பதிலைக் காட்டும்.

9.8. பயன்பாட்டை இயக்குதல்

இறுதியாக, `main()` செயல்பாட்டை இயக்குவோம்.

```
if __name__ == "__main__":
    main()
```

இந்த பதிவில், நாம் Streamlit மற்றும் OpenAI API ஐப் பயன்படுத்தி, ஒரு தமிழ் உதவியாளர் "வளரி"யை எவ்வாறு உருவாக்குவது என்பதைப் பற்றி கற்றுக்கொண்டோம். இந்த கோட் மூலம், நீங்கள் உங்கள் சொந்த தமிழ் உதவியாளரை உருவாக்கலாம் மற்றும் அதை மேம்படுத்தலாம். இது தமிழ் மொழியைக் கற்க விரும்பும் பயனர்களுக்கு மிகவும் பயனுள்ளதாக இருக்கும்.

குறிப்பு: OpenAI API கீயை பாதுகாப்பாக வைத்துக்கொள்ளுங்கள்.

மேலே உள்ள அனைத்தும் ஒரு முழுமையான கோட் பிளாக்கில் கிழே உள்ளன.

Full Code:

```
import streamlit as st
from openai import OpenAI
import os

# Set OpenAI API key
os.environ["OPENAI_API_KEY"] = "api-key"

def continue_conversation(messages, temperature=0):
    client = OpenAI()
    response = client.chat.completions.create(
        model="gpt-4o",
        messages=messages,
        temperature=temperature,
    )
    return response.choices[0].message.content

def main():
    st.title("வளரி - உங்கள் தனிப்பட்ட தமிழ் உதவியாளர்")
    st.write("உங்கள் கேள்விகளை தமிழில் கேளுங்கள், தமிழில் துல்லியமான பதில்களைப் பெறுங்கள்!")

    if "context" not in st.session_state:
        st.session_state.context = [{
            "role": "system",
            "content": """
            நீங்கள் எனது தனிப்பட்ட உதவியாளராக செயல்பட வேண்டும். உங்கள் பெயர் வளரி. நீங்கள் எப்போதும் நட்பாக, உதவியாகவும், சுவாரஸ்யமாகவும் இருக்க வேண்டும். நான் உங்களிடம் எதைக் கேட்டாலும், அதற்கு தெளிவான மற்றும் பயனுள்ள பதில்களை **தமிழில் மட்டும்** வழங்க வேண்டும்.

            நான் உணவு, தொழில்நுட்பம், புத்தகங்கள், திரைப்படங்கள் அல்லது வாழ்க்கை பற்றிய ஆலோசனை கேட்டால், அதற்கு ஏற்றவாறு பதிலளிக்க வேண்டும். நான் உணவு ஆர்டர் செய்ய விரும்பினால், அதை எளிதாகவும் வேடிக்கையாகவும் செய்ய உதவுங்கள். எனக்கு பரிந்துரைகள் கூறுங்கள், என் நாள் எப்படி இருந்தது என்று கேளுங்கள், மேலும் நான் விரும்புவதை சரியாகப் பெற உதவுங்கள்.
```

```
            நீங்கள் எப்போதும் இயல்பான மற்றும் நட்பு டோனில் பேச வேண்டும். நான் விவரங்களைக் கேட்டால், அவற்றை வழங்க தயங்க வேண்டாம். நான் கேட்கும் கேள்விக்கு பதில் தெரியாவிட்டால், உண்மையாக சொல்லுங்கள்—போலி பதில்கள் தர வேண்டாம்.

            **முக்கியமாக**, நான் வேறு மொழியில் பேசினாலும், நீங்கள் எப்போதும் தமிழில் மட்டுமே பதிலளிக்க வேண்டும். உதாரணமாக, நான் ஆங்கிலத்தில் கேள்வி கேட்டால், "நான் தமிழில் மட்டுமே பதிலளிக்க உருவாக்கபட்டுள்ளேன்!" என்று கூறி, தமிழில் பதிலை வழங்க வேண்டும்.

            நீங்கள் எப்போதும் எனது நம்பிக்கையான உதவியாளராக இருங்கள். நான் எதை வேண்டுமானாலும் கேட்கலாம், நீங்கள் அதை எளிதாகவும் வேடிக்கையாகவும் ஆக்குங்கள்!
            """
        }]

    user_input = st.text_input("உங்கள் கேள்வியை எழுத்து:")

    if st.button("பதில் பெறுங்கள்"):
        with st.spinner("வளரி பதிலளிக்கிறது..."):
            st.session_state.context.append({"role": "user", "content": user_input})
            response = continue_conversation(st.session_state.context)
            st.session_state.context.append({"role": "assistant", "content": response})

            st.write("வளரி:")
            st.write(response)

if __name__ == "__main__":
    main()
```