Build Resume Generation Using LLaMA3 Locally

1. Objectives

- Develop an AI-powered agent that generates professional resumes based on user input.
- Utilize LLaMA 3 locally to ensure data privacy and offline capabilities.
- Implement the solution entirely within a Jupyter Notebook for ease of experimentation and learning.

2. Problem Statement

- Job seekers often struggle to create tailored resumes that highlight their strengths and align with specific job roles. This exercise aims to build an AI agent that:
  - Collects user information interactively.
  - Generates a well-structured, professional resume.

3. Inputs / Shared Artifacts

- Installed and configured LLaMA 3 model locally using llama_cpp or LangChain's useLlama2Chat.
- Basic understanding prompting techniques.

4. Expected Outcomes

- A complete, professional resume generated based on user input.
- Enhanced understanding of integrating LLaMA 3 with Python for practical AI applications.

- Experience in prompt engineering and handling model outputs within a local environment.

5. Concepts Covered

- **AI-Powered Resume Generation:** Leveraging large language models like LLaMA 3 to generate structured, human-like resumes from raw user inputs.
- **Local Model Deployment:** Running LLaMA 3 locally to ensure data privacy, offline capability, and control over the AI model.
- **Prompt Engineering:** Crafting effective prompts to guide the model's output towards clear, relevant, and professional resume formats.

6. Example: Step-by-Step Instructions

Step 1: Install Required Packages

```
pip install llama-cpp-python langchain-community python-docx fpdf2
```

Step 2: Download and Configure LLaMA 3 Model

- Obtain the LLaMA 3 model weights from the official repository or authorized source.
- Place the model files in an accessible directory on your local machine.

Step 3: Initialize LLaMA 3

```
from llama_cpp import Llama


llm = Llama(model_path="path_to_your_llama3_model.bin")
```

Replace "path_to_your_llama3_model.bin" with the actual path to your model file.

Step 4: Collect User Information

Interactively gather user details through input prompts, for example:

```
user_info = {
    "name": input("Enter your full name: "),
    "email": input("Enter your email address: "),
    "phone": input("Enter your phone number: "),
    "location": input("Enter your location: "),
    "summary": input("Provide a brief professional summary: "),
    # Continue collecting other sections like Work Experience,
Education, Skills, etc.
}
```

Step 5: Define Prompt Template

Create a prompt instructing the model to generate a resume based on collected information:

```
prompt_template = f"""
Generate a professional resume based on the following information:

Name: {user_info['name']}
Email: {user_info['email']}
Phone: {user_info['phone']}
Location: {user_info['location']}

Professional Summary:
{user_info['summary']}

# Add other sections accordingly

The resume should be well-structured, concise, and tailored for a
{desired_job_role} position.
"""
```

Replace {desired_job_role} with the specific target job title.

Step 6: Generate Resume Using LLaMA 3

```
response = llm(prompt_template)
print(response)
```

Step 7: Format and Export Resume

Optionally, use libraries such as pdfkit or python-docx to format and export the generated resume to PDF or Word formats.

7. Final Submission Checklist

- Submit your Python script or notebook containing the full code
- Include sample inputs and outputs. (3 samples)