GENERATIVE AI AND ITS APPLICATIONS - UE23CS342BA4

Hands-on 2

Assignment

NAME : GREESHMA YP
SRN : PES2UG22CS207
SEM : 6
SEC : A
DATE : 25-01-2026

Mixture of Experts (MoE) Router – Groq API

Cell 1 – Package Installation

This cell installs the required libraries, namely groq and python-dotenv. Its main purpose is to prepare the Google Colab environment with the necessary dependencies. These libraries are essential for connecting to the Groq API and for securely handling environment variables. By executing this step, the notebook becomes fully equipped to communicate with the Groq language model and manage API keys safely.

Cell 2 – Secure API Key Setup

This cell imports the modules os, getpass, and Groq, then securely prompts the user to enter the Groq API key. The getpass() function ensures that the API key is hidden during input so it does not appear in the notebook output. The key is stored in the environment variable GROQ_API_KEY, after which a Groq client instance is initialized using this key. This approach guarantees secure authentication and prevents accidental exposure of sensitive credentials while establishing a connection with the Groq API.

Cell 3 – Define MODEL_CONFIG (Expert Definitions)

This cell defines a configuration dictionary named MODEL_CONFIG, which contains multiple expert roles within the system. The defined experts include a Technical Expert, a Billing Expert, a General Expert, and a Tool Expert for bonus functionality. Each expert configuration contains a system prompt and a temperature value. Although all experts use the same base language model, their behavior di ers because each one operates under a di erent system prompt and temperature setting. This design enables the implementation of a Mixture of Experts architecture by simulating specialized roles within a single model framework.

Cell 4 – Router Function (route_prompt)

This function is responsible for classifying the user's input into a specific category. It uses a language model to determine whether the query belongs to the technical, billing, general, or tool category. The temperature is set to 0 to ensure consistent and deterministic classification results. The function receives the user query, sends a structured classification prompt to the model, and returns only the category name without any additional explanation. This function serves as the core routing mechanism in the Mixture of Experts system by identifying user intent and directing the request to the appropriate expert.

Cell 5 – Tool Function (get_bitcoin_price)

This cell implements the bonus tool functionality. Instead of using the language model to generate a response, this function directly returns a mock Bitcoin price. The purpose of this design is to demonstrate how external tools can be integrated into the architecture. When the router detects a financial data query, such as a request for the current price of Bitcoin, the system routes the request to this function instead of the language model. This shows that the MoE system can extend beyond pure text generation and incorporate external functionalities.

Cell 6 – Orchestrator Function (process_request)

This is the main control function of the system and acts as the orchestration layer of the architecture. First, it calls the route_prompt() function to determine the category of the user input. Once the category is identified, the function checks whether it corresponds to the tool expert. If so, it directly calls the tool function. Otherwise, it retrieves the appropriate system prompt from the MODEL_CONFIG dictionary and sends both the system message and the user input to the language model. The expert-specific temperature value is applied during this process. Finally, the function returns the generated response. This function e ectively connects the routing logic with expert selection and response generation, completing the Mixture of Experts workflow.

Cell 7 – Test Cases

This cell is used to evaluate the behavior of the entire system using multiple types of queries. It includes examples such as a technical programming error, a billing-related refund request, a general conversational greeting, and a Bitcoin price query for testing the tool functionality. These tests demonstrate that the router correctly classifies user input, the appropriate expert is selected, the tool function executes when required, and the overall end-to-end Mixture of Experts pipeline works as intended.

This final validation confirms that the architecture operates correctly across different scenarios.