Installing Dependencies

```
!pip install transformers accelerate bitsandbytes
!pip install langchain==1.0.3
!pip install langchain-community
```

Show hidden output

```
from huggingface_hub import login

login("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")
```

LLM

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig, pipeline
from langchain_community.llms import HuggingFacePipeline # Corrected import


model_id = "meta-llama/Meta-Llama-3-8B-Instruct"

quantization_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_compute_dtype=torch.bfloat16,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4"
)


tokenizer = AutoTokenizer.from_pretrained(model_id)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    quantization_config=quantization_config,
    device_map="auto"
)

print("Model loaded successfully!")


pipe = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    max_new_tokens=256,
    temperature=0.7,
    do_sample=True
)

llm = HuggingFacePipeline(pipeline=pipe)

print("LLM wrapper ready!")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Loading checkpoint shards: 100%                                    4/4 [00:17<00:00, 3.58s/it]
Device set to use cuda:0
Model loaded successfully!
LLM wrapper ready!
/tmp/ipython-input-2958176007.py:35: LangChainDeprecationWarning: The class `HuggingFacePipeline` was deprecated in LangChai
  llm = HuggingFacePipeline(pipeline=pipe)
```

```
prompt = "Explain the cause of Allergy"

inputs = tokenizer(prompt, return_tensors="pt").to(model.device)
```

```
outputs = model.generate(
    **inputs,
    max_new_tokens=50,
    do_sample=True,
    temperature=0.7
)

response = tokenizer.decode(outputs[0], skip_special_tokens=True)
print("LLM Response:")
print(response)
```

```
Setting `pad_token_id` to `eos_token_id`:128001 for open-end generation.
LLM Response:
Explain the cause of Allergy.
An allergy is an overreaction of the immune system to a harmless substance, such as pollen, dust, or certain foods. The immu
```

Tools

```
import requests

def pubmed_search(query: str) -> str:
    """
    Search PubMed for biomedical papers and return summaries.
    """
    search_url = "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi"
    search_params = {
        "db": "pubmed",
        "term": query,
        "retmax": 5,
        "retmode": "json"
    }
    search_response = requests.get(search_url, params=search_params).json()

    id_list = search_response.get("esearchresult", {}).get("idlist", [])
    if not id_list:
        return "No research papers found for your query."

    summary_url = "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esummary.fcgi"
    summary_params = {
        "db": "pubmed",
        "id": ",".join(id_list),
        "retmode": "json"
    }
    summary_response = requests.get(summary_url, params=summary_params).json()

    result_summaries = []
    for paper_id in id_list:
        doc = summary_response.get("result", {}).get(paper_id, {})
        title = doc.get("title", "No title found")
        source = doc.get("source", "Unknown source")
        pubdate = doc.get("pubdate", "No date")
        result_summaries.append(f"{title} ({source}, {pubdate})")

    return "Top papers:\n" + "\n".join(result_summaries)


# Test the plain function directly
query = "brain tumor classification"
result = pubmed_search(query)
print(result)
```

```
Top papers:
Effectiveness of Infliximab for Refractory Nonischemic Cerebral Enhancing Foreign-Body Granulomatous Lesions After Endovascu
Integrated transcriptomic landscape of medulloblastoma and ependymoma reveals novel tumor subtype-specific biology. (Neuro C
An integrated analysis of three medulloblastoma clinical trials refines risk-stratification approaches for reducing toxicity
Diagnostic challenges of gliosarcoma: case report of a rare glioblastoma histopathological variant. (Front Radiol, 2025)
Genetic Markers and Mutations in Primary Spinal Cord Tumors and Their Impact on Clinical Management. (Brain Sci, 2025 Sep 23
```

```
import requests

def openfda_search(query: str) -> str:
    """
    Search OpenFDA for drug information by name or keyword.
    """
    base_url = "https://api.fda.gov/drug/label.json"
    params = {
        "search": query,
        "limit": 5
    }
    try:
        response = requests.get(base_url, params=params)
        data = response.json()
        results = data.get("results", [])
        if not results:
            return "No drug info found for your query."
        summaries = []
        for res in results:
            openfda = res.get("openfda", {})
            brand_name = openfda.get("brand_name", ["Unknown brand"])[0]
            manufacturer = openfda.get("manufacturer_name", ["Unknown manufacturer"])[0]
            purpose = res.get("purpose", ["No purpose info"])[0]
            summaries.append(f"{brand_name} by {manufacturer}: {purpose}")
        return "Drugs found:\n" + "\n".join(summaries)
    except Exception as e:
        return f"OpenFDA request failed: {str(e)}"


def clinical_tables_search(query: str) -> str:
    """
    Search Clinical Tables API for clinical condition info.
    Replace base_url with the correct clinical tables API endpoint.
    """
    base_url = "https://clinicaltables.nlm.nih.gov/api/conditions/v3/search"
    params = {
        "terms": query,
        "maxList": 5
    }
    try:
        response = requests.get(base_url, params=params)
        data = response.json()
        results = data[3]  # List of clinical conditions returned
        if not results:
            return "No clinical data found for your query."
        summaries = [f"{item[1]} ({item[2]})" for item in results]
        return "Clinical conditions found:\n" + "\n".join(summaries)
    except Exception as e:
        return f"ClinicalTables API request failed: {str(e)}"
```

```
print(openfda_search("aspirin"))
print(clinical_tables_search("diabetes"))
```

```
Drugs found:
Naproxen by A-S Medication Solutions: No purpose info
Glimepiride by American Health Packaging: No purpose info
Unknown brand by Unknown manufacturer: Purpose Pain reliever/fever reducer
Pain Reliever Extra Strength by Valu Merchandisers Company: Purpose Pain reliever/fever reducer
Unknown brand by Unknown manufacturer: No purpose info
ClinicalTables API request failed: list index out of range
```