📚 Aula 3 – Chatbot com LLM (Hugging Face)

Objetivo: conectar o chatbot que vocês já têm a um modelo de linguagem (LLM) para gerar respostas mais naturais.

---

1. O que é uma LLM?

- Modelo de linguagem treinado em grandes volumes de texto.
- Pode gerar respostas, completar frases ou até resumir texto.
- Baseia-se em padrões de linguagem aprendidos durante o treinamento para prever a próxima palavra ou frase.

2. Hugging Face Hub

- Plataforma com centenas de modelos pré-treinados.
- Acesso via **Transformers** (biblioteca Python) ou **Inference API** (API pronta para uso).
- Permite testar, treinar e integrar modelos de forma rápida.

3. Integração com chatbot

- Em vez de respostas fixas, a API do chatbot chama o modelo ML e devolve a resposta gerada.
- Conceito de **"prompt"** → texto enviado ao modelo para obter a resposta desejada.
- A forma como o prompt é escrito pode alterar drasticamente a qualidade e relevância da resposta.

4. Prompt Engineering

- Técnica de criar, ajustar e otimizar prompts para melhorar a resposta do modelo.
- Ex.: "Resuma este texto em 3 frases" vs. "Explique este texto como se eu tivesse 5 anos".
- Essencial para LLMs, pois a saída depende muito de **como a pergunta é feita**.

5. RAG (Retrieval-Augmented Generation)

- Combina **recuperação de informação** (retrieval) com **geração de texto** (generation).
- O modelo não depende apenas do que foi treinado; ele busca informações externas (documentos, bases de dados) e depois gera a resposta.
- Vantagem: respostas mais precisas e atualizadas, mesmo para assuntos que a LLM não viu durante o treinamento.

ATIVIDADE 1

Montar um exemplo em Python de RAG que conecte no Hugging Face.

◆ Prática: Integração com Hugging Face

---

requirements.txt

```
fastapi
uvicorn
sentence-transformers
faiss-cpu
requests
```

---

app.py

```
from fastapi import FastAPI
from pydantic import BaseModel
from sentence_transformers import SentenceTransformer
import faiss
import numpy as np
import requests
import os

# ------------------------
# Config simples
# ------------------------
HF_TOKEN = os.getenv("HUGGINGFACEHUB_API_TOKEN")  # exporte no
terminal
HF_MODEL = "TinyLlama/TinyLlama-1.1B-Chat-v1.0"   # pode trocar

# Modelo de embeddings
embedder = SentenceTransformer("sentence-transformers/all-MiniLM-L6-
v2")
dim = 384
index = faiss.IndexFlatIP(dim)  # inner product
documents = []  # guardamos os textos aqui

# ------------------------
# FastAPI
# ------------------------
app = FastAPI()

class Ingest(BaseModel):
    text: str

class Ask(BaseModel):
    question: str

# Endpoint para adicionar texto
@app.post("/ingest")
def ingest(item: Ingest):
    vec = embedder.encode([item.text], convert_to_numpy=True,
normalize_embeddings=True)
    index.add(vec)
    documents.append(item.text)
    return {"status": "added", "text": item.text}
```

```
# Endpoint para perguntar
@app.post("/ask")
def ask(item: Ask):
    # Embedding da pergunta
    q_vec = embedder.encode([item.question], convert_to_numpy=True,
normalize_embeddings=True)
    D, I = index.search(q_vec, 1)  # pega o mais parecido
    context = documents[I[0][0]] if len(documents) > 0 else "Sem
contexto disponível."
    # Monta prompt simples
    prompt = f"Contexto: {context}\n\nPergunta:
{item.question}\nResposta:"
    # Chamada ao Hugging Face Inference API
    headers = {"Authorization": f"Bearer {HF_TOKEN}"}
    payload = {"inputs": prompt, "parameters": {"max_new_tokens":
200}}
    resp = requests.post(f"https://api-
inference.huggingface.co/models/{HF_MODEL}",
                         headers=headers, json=payload)
    data = resp.json()
    answer = data[0]["generated_text"] if isinstance(data, list) else
str(data)
    return {"answer": answer, "context": context}
```

Como rodar

1. Instale as dependências:
    1. `pip install -r requirements.txt`
2. Configure sua chave do Hugging Face:
    1. `export HUGGINGFACEHUB_API_TOKEN=hf_xxx...`
3. Rode a API:
    1. `Python –m uvicorn app:app --reload --port 8000`

Testando

1. **Adicionar um texto:**
2. `curl -X POST http://localhost:8000/ingest \`
3. `  -H "Content-Type: application/json" \`
4. `  -d '{"text": "A FastAPI é um framework moderno em Python para criar APIs de forma rápida."}'`
5. **Perguntar:**
6. `curl -X POST http://localhost:8000/ask \`
7. `  -H "Content-Type: application/json" \`
8. `  -d '{"question": "O que é FastAPI?"}'`

◆ Pontos de discussão

- Latência e recursos: modelos grandes podem ser lentos.
- Custo: Hugging Face oferece API gratuita limitada, mas o modelo local consome GPU/CPU.