```
!pip install sentence-transformers faiss-cpu transformers duckduckgo-search
```

```
Requirement already satisfied: sentence-transformers in /usr/local/lib/python3.11/dist-packages (4.1.0)
Collecting faiss-cpu
  Downloading faiss_cpu-1.11.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (4.8 kB)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.52.4)
Collecting duckduckgo-search
  Downloading duckduckgo_search-8.0.4-py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (2.6.0+cu124)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.15.3)
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (0.32.4
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (11.2.1)
Requirement already satisfied: typing_extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.14.
Requirement already satisfied: numpy<3.0,>=1.25.0 in /usr/local/lib/python3.11/dist-packages (from faiss-cpu) (2.0.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from faiss-cpu) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.18.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21.1)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: click>=8.1.8 in /usr/local/lib/python3.11/dist-packages (from duckduckgo-search) (8.2.1)
Collecting primp>=0.15.0 (from duckduckgo-search)
  Downloading primp-0.15.0-cp38-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Requirement already satisfied: lxml>=5.3.0 in /usr/local/lib/python3.11/dist-packages (from duckduckgo-search) (5.4.0)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-tra
Requirement already satisfied: hf-xet<2.0.0,>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3.1.6)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-trans
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-tra
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (3
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers) (1
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch>=1.11.0->sente
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.4.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
```

```
!pip install duckduckgo-search
```

```
Requirement already satisfied: duckduckgo-search in /usr/local/lib/python3.11/dist-packages (8.0.4)
Requirement already satisfied: click>=8.1.8 in /usr/local/lib/python3.11/dist-packages (from duckduckgo-search) (8.2.1)
Requirement already satisfied: primp>=0.15.0 in /usr/local/lib/python3.11/dist-packages (from duckduckgo-search) (0.15.0)
Requirement already satisfied: lxml>=5.3.0 in /usr/local/lib/python3.11/dist-packages (from duckduckgo-search) (5.4.0)
```

```
from duckduckgo_search import DDGS
```

```
def web_search(query):
    with DDGS() as ddgs:
        results = list(ddgs.text(query, max_results=3))
    if results:
        return results[0]["body"] if "body" in results[0] else results[0]["href"]
    else:
```

```
        return "Bro, net lo kuda information kanipinchaledu "
```

```
print(web_search("Who is the current CEO of Google?"))
```

```
Pichai Sundararajan (born June 10, 1972), better known as Sundar Pichai (pronounced: / ˈ s ʊ n d ɜːr p ɪ ˈ tʃ eɪ /), is an American busi
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Chat.json to Chat.json
```

Start coding or generate with AI.

```
# Basic Imports
import json
import numpy as np
from sentence_transformers import SentenceTransformer
import faiss
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
from duckduckgo_search import DDGS


# Web search function
def web_search(query):
    with DDGS() as ddgs:
        results = list(ddgs.text(query, max_results=3))
    if results:
        return results[0]["body"] if "body" in results[0] else results[0]["href"]
    else:
        return "Bro, net lo kuda information kanipinchaledu"


# Load chat messages
def load_messages(path):
    with open(path, "r", encoding="utf-8") as f:
        data = json.load(f)
    messages = [entry["message"] for entry in data if "message" in entry and entry["sender"] == "Sarfaraz Ahamad"]
    return messages
#sarfaraz Ahamad is my friend where he helped me to collect the customized dataset of friendly tone

# Embed messages and build FAISS index
def build_index(messages):
    encoder = SentenceTransformer("all-MiniLM-L6-v2")
    embeddings = encoder.encode(messages, show_progress_bar=True)
    index = faiss.IndexFlatL2(embeddings.shape[1])
    index.add(np.array(embeddings))
    return encoder, index, embeddings

# Get similar responses
def retrieve_similar(query, encoder, index, messages, k=3):
    query_vec = encoder.encode([query])
    distances, indices = index.search(query_vec, k)
    return [messages[i] for i in indices[0]]

# Load open LLM
def load_llm():
    tokenizer = AutoTokenizer.from_pretrained("mistralai/Mistral-7B-Instruct-v0.1", use_auth_token=True)
    model = AutoModelForCausalLM.from_pretrained("mistralai/Mistral-7B-Instruct-v0.1", device_map="auto")
    return pipeline("text-generation", model=model, tokenizer=tokenizer, max_new_tokens=100)

# Final response generator
def generate_reply(query, encoder, index, messages, llm_pipeline):
    sims = retrieve_similar(query, encoder, index, messages)
    if sims and sims[0]:
        return f"Personal touch 🍇:\n{sims[0]}"
    else:
        # Web fallback
        web_info = web_search(query)
        if web_info:
            return f"Google se mila info 🌐:\n{web_info}"
```

```
        # LLM fallback
        response = llm_pipeline(f"Tu kya bolega jab koi puche: {query}")
        return "Soch ke bola bhai:\n" + response[0]['generated_text']

"Ready to load your file and build the pipeline!"
```

```
'Ready to load your file and build the pipeline!'
```

```
import json

def load_messages(path="Chat.json"):
    with open(path, "r", encoding="utf-8") as f:
        data = json.load(f)
    messages = [d["message"] for d in data if "message" in d]
    return messages

messages = load_messages()
print("Sample:", messages[:4])
```

```
Sample: ['Bochulo photo tisi nak pettadaniki baleda?', 'Nak avasaram ledu anna', 'Inkot pettu', 'Ninnu avad adagala']
```

```
from sentence_transformers import SentenceTransformer
import numpy as np

encoder = SentenceTransformer("all-MiniLM-L6-v2")
message_embeddings = encoder.encode(messages, show_progress_bar=True)
```

```
/usr/local/lib/python3.11/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 it as secre
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(
modules.json: 100%  349/349 [00:00<00:00, 5.04kB/s]
config_sentence_transformers.json: 100%  116/116 [00:00<00:00, 1.93kB/s]
README.md: 100%  10.5k/10.5k [00:00<00:00, 195kB/s]
sentence_bert_config.json: 100%  53.0/53.0 [00:00<00:00, 2.38kB/s]
config.json: 100%  612/612 [00:00<00:00, 8.13kB/s]
model.safetensors: 100%  90.9M/90.9M [00:01<00:00, 74.5MB/s]
tokenizer_config.json: 100%  350/350 [00:00<00:00, 32.1kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 4.92MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 23.4MB/s]
special_tokens_map.json: 100%  112/112 [00:00<00:00, 8.90kB/s]
config.json: 100%  190/190 [00:00<00:00, 13.4kB/s]
Batches: 100%  30/30 [00:08<00:00, 9.69it/s]
```

```
import faiss

dimension = message_embeddings.shape[1]
index = faiss.IndexFlatL2(dimension)
index.add(np.array(message_embeddings))
```

```
def get_similar_messages(query, k=3):
    query_embedding = encoder.encode([query])
    distances, indices = index.search(query_embedding, k)
    return [messages[i] for i in indices[0]]
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

model_id = "HuggingFaceH4/zephyr-7b-alpha"
```

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

generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

```
tokenizer_config.json: 100%  1.43k/1.43k [00:00<00:00, 108kB/s]
tokenizer.model: 100%  493k/493k [00:00<00:00, 1.09MB/s]
tokenizer.json: 100%  1.80M/1.80M [00:00<00:00, 20.2MB/s]
added_tokens.json: 100%  42.0/42.0 [00:00<00:00, 2.88kB/s]
special_tokens_map.json: 100%  168/168 [00:00<00:00, 15.0kB/s]
config.json: 100%  628/628 [00:00<00:00, 56.1kB/s]
model.safetensors.index.json: 100%  23.9k/23.9k [00:00<00:00, 1.97MB/s]
Fetching 8 files: 100%  8/8 [14:01<00:00, 348.01s/it]
model-00001-of-00008.safetensors: 100%  1.89G/1.89G [13:53<00:00, 6.51MB/s]
model-00005-of-00008.safetensors: 100%  1.98G/1.98G [14:01<00:00, 5.61MB/s]
model-00008-of-00008.safetensors: 100%  816M/816M [09:08<00:00, 2.19MB/s]
model-00007-of-00008.safetensors: 100%  1.98G/1.98G [12:50<00:00, 4.28MB/s]
model-00003-of-00008.safetensors: 100%  1.98G/1.98G [14:00<00:00, 3.88MB/s]
model-00006-of-00008.safetensors: 100%  1.95G/1.95G [13:56<00:00, 7.93MB/s]
model-00004-of-00008.safetensors: 100%  1.95G/1.95G [13:51<00:00, 5.82MB/s]
model-00002-of-00008.safetensors: 100%  1.95G/1.95G [14:01<00:00, 5.89MB/s]
Loading checkpoint shards: 100%  8/8 [00:20<00:00, 10.20s/it]
generation_config.json: 100%  111/111 [00:00<00:00, 5.16kB/s]
WARNING:accelerate.big_modeling:Some parameters are on the meta device because they were offloaded to the cpu and disk.
Device set to use cpu
```

```
from duckduckgo_search import DDGS

def web_search(query):
    with DDGS() as ddgs:
        results = list(ddgs.text(query, max_results=3))
    if results:
        return results[0]["body"] if "body" in results[0] else results[0]["href"]
    else:
        return "Bro, net lo kuda information kanipinchaledu"



def rewrite_in_sarfaraz_style(answer):
    """Rewrites any factual or generated response in Sarfaraz-style slang."""
    style_prompt = f"""
    Convert the following answer into casual Hindi-English slang like a funny and emotionally supportive friend (Sarfaraz style):

    Original: {answer}
    Styled:
    """
    response = generator(style_prompt, max_new_tokens=60, do_sample=True, temperature=0.9)[0]["generated_text"]
    return response.split("Styled:")[-1].strip()



def generate_response(user_input):
    similar = get_similar_messages(user_input)
    if similar:
        return f"Sarfaraz style lo cheppalante: {similar[0]}"

    # Web search fallback
    if any(q in user_input.lower() for q in ["what", "who", "when", "where", "how", "latest", "news"]):
        web_ans = web_search(user_input)
        return "Sarfaraz style lo cheppalante: " + rewrite_in_sarfaraz_style(web_ans)

    # Fallback to LLM and stylize
    prompt = f"Answer this in a factual way: {user_input}"
    raw_reply = generator(prompt, max_new_tokens=60, do_sample=True, temperature=0.7)[0]["generated_text"]
```

```
    return "Sarfaraz style lo cheppalante: " + rewrite_in_sarfaraz_style(raw_reply)
```

```
print(generate_response("Why are you ignoring me? 😳"))
```

```
Sarfaraz style lo cheppalante: And I am mad about you 🙄
```

Start coding or generate with AI.