1. Setup Google Colab Environment

```
# Install necessary libraries
!pip install transformers
!pip install datasets
!pip install sentence-transformers
!pip install nltk
```

---

2. Basic Chatbot Using Pre-trained Transformer (e.g., DialoGPT)

```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# Load pre-trained DialoGPT model
tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-small")
model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-small")

# Chat loop
print("Chatbot is ready! Type 'quit' to exit.")
chat_history_ids = None

while True:
    user_input = input("You: ")
    if user_input.lower() == 'quit':
        break

    new_input_ids = tokenizer.encode(user_input + tokenizer.eos_token,
return_tensors='pt')

    bot_input_ids = torch.cat([chat_history_ids, new_input_ids], dim=-1) if
chat_history_ids is not None else new_input_ids
    chat_history_ids = model.generate(bot_input_ids, max_length=1000,
pad_token_id=tokenizer.eos_token_id)

    response = tokenizer.decode(chat_history_ids[:, bot_input_ids.shape[-
1]:][0], skip_special_tokens=True)
    print("Bot:", response)
```

---

3. Loading FAQ Data for Contextual Support

You can improve the chatbot by feeding it a dataset of FAQs or customer queries.

```
from datasets import load_dataset

# Example: Load a sample FAQ dataset (replace with your own CSV)
```

```
faq_data = {
    "questions": ["How do I reset my password?", "What is the return
policy?"],
    "answers": ["To reset your password, click on 'Forgot password' link on
the login page.",
                "You can return your product within 30 days of purchase."]
}
```

---

4. Add Semantic Search for Contextual Matching

```
from sentence_transformers import SentenceTransformer, util

model_embed = SentenceTransformer('all-MiniLM-L6-v2')

faq_questions = faq_data['questions']
faq_answers = faq_data['answers']
faq_embeddings = model_embed.encode(faq_questions, convert_to_tensor=True)

def get_answer(user_query):
    query_embedding = model_embed.encode(user_query, convert_to_tensor=True)
    scores = util.pytorch_cos_sim(query_embedding, faq_embeddings)
    best_idx = torch.argmax(scores)
    return faq_answers[best_idx]

# Test
print(get_answer("How can I change my password?"))
```

---

5. Combine Chatbot + FAQ Response

```
def intelligent_chat(user_query):
    # Try FAQ matching first
    answer = get_answer(user_query)
    if answer:
        return answer

    # Fallback to chatbot model
    input_ids = tokenizer.encode(user_query + tokenizer.eos_token,
return_tensors='pt')
    output_ids = model.generate(input_ids, max_length=1000,
pad_token_id=tokenizer.eos_token_id)
    return tokenizer.decode(output_ids[:, input_ids.shape[-1]:][0],
skip_special_tokens=True)

# Chat loop
while True:
```

```
    query = input("You: ")
    if query.lower() == 'quit':
        break
    print("Bot:", intelligent_chat(query))
```

---

6. Optional: Add GUI using Gradio

```
!pip install gradio

import gradio as gr

def chatbot_interface(user_input):
    return intelligent_chat(user_input)

gr.Interface(fn=chatbot_interface, inputs="text", outputs="text",
title="Customer Support Chatbot").launch()
```

Improved & Attractive Chatbot in Google Colab

7. Install Dependencies

```
!pip install transformers sentence-transformers gradio datasets
```

---

8. Import Libraries

```
from transformers import AutoModelForCausalLM, AutoTokenizer
from sentence_transformers import SentenceTransformer, util
import torch
import gradio as gr
```

---

9. Load Models

```
# Load chatbot model (DialoGPT)
chat_tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-medium")
chat_model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-
medium")

# Load sentence transformer for FAQ matching
embedder = SentenceTransformer('all-MiniLM-L6-v2')
```

---

10. Define FAQ Knowledge Base

```
faq_data = {
    "questions": [
        "How do I reset my password?",
```

```
        "What is your return policy?",
        "How can I contact customer support?",
        "How do I track my order?",
        "What payment methods are accepted?"
    ],
    "answers": [
        "To reset your password, click 'Forgot password' on the login page.",
        "We accept returns within 30 days with the original receipt.",
        "You can reach support via email or the contact form on our site.",
        "You can track your order using the tracking link sent to your
email.",
        "We accept credit cards, debit cards, PayPal, and Apple Pay."
    ]
}

faq_embeddings = embedder.encode(faq_data['questions'],
convert_to_tensor=True)
```

---

11. Define Smart Chat Function

```
def chatbot_response(user_input, history=[]):
    # Semantic search in FAQs
    query_embedding = embedder.encode(user_input, convert_to_tensor=True)
    scores = util.pytorch_cos_sim(query_embedding, faq_embeddings)
    best_score = torch.max(scores).item()
    best_idx = torch.argmax(scores).item()

    # Threshold for FAQ match confidence
    if best_score > 0.7:
        return faq_data['answers'][best_idx]

    # Otherwise, fallback to DialoGPT
    new_input_ids = chat_tokenizer.encode(user_input +
chat_tokenizer.eos_token, return_tensors='pt')

    if history:
        bot_input_ids = torch.cat([history[-1], new_input_ids], dim=-1)
    else:
        bot_input_ids = new_input_ids

    chat_history_ids = chat_model.generate(bot_input_ids, max_length=1000,
pad_token_id=chat_tokenizer.eos_token_id)

    response = chat_tokenizer.decode(chat_history_ids[:,
bot_input_ids.shape[-1]:][0], skip_special_tokens=True)

    # Keep history for context
    history.append(chat_history_ids)
    return response
```

---

12. Build GUI with Gradio

```
def gradio_chat(user_input, chat_history=[]):
    response = chatbot_response(user_input, chat_history)
```

```
    chat_history.append((user_input, response))
    return "", chat_history

chat_ui = gr.ChatInterface(
    fn=gradio_chat,
    title="Customer Support Chatbot",
    theme="compact",
    chatbot=gr.Chatbot(height=400),
    textbox=gr.Textbox(placeholder="Ask your question...", lines=2),
    clear_btn="Clear",
    submit_btn="Send"
)

chat_ui.launch()
```
