policy_chatbot

September 28, 2025

0.0.1 Nestle Policy Chatbot - Simplilearn Submission

- **Name:** Shrawanika Wakde
- **Email:** shrawanika@gmail.com
- **Program:** Advanced Executive Program in Applied Generative AI
- **Batch:** IITM AG June 2025 Cohort 1
- **Date:** 2025-09-28

1 Project

Task: Your task is to develop a conversational chatbot. This chatbot must answer queries about Nestlé's HR reports efficiently. Use Python libraries, OpenAI's GPT model, and Gradio UI. These tools will help you create a user-friendly interface. This interface will extract and process information from documents. It will provide accurate responses to user queries.

Action: • Import essential tools and set up OpenAI's API environment. • Load Nestle's HR policy using PyPDFLoader and split it for easy processing. • Create vector representations for text chunks using Chroma dB and OpenAI's embeddings. • Build a question-answering system using the GPT-3.5 Turbo model to retrieve answers from text chunks. • Create a prompt template to guide the chatbot in understanding and responding to users. • Use Gradio to build a user-friendly chatbot interface, enabling interaction and information retrieval.

1.1 Challenges Encountered

1. **Using static file input instead of UI upload**

- Currently, the system relies on static PDF files rather than allowing users to upload documents through the UI.
- Because of Gradio instability in JupyterLab, not using Gradio Upload Button which uses Blocks
- Using Gradio Blocks sometimes causes random errors, making the interface unreliable in the lab environment.

2. **Using FAISS instead of Chroma** :

- ChromaDB requires SQLite, but the lab environment has an outdated, unsupported SQLite version.
- This prevents creation of both in-memory and persistent vector stores.
- Upgrading SQLite requires admin access, which is unavailable in the lab.
- Attempts to use duckdb+parquet as a workaround fail because the installed Chroma version still depends on SQLite internally.

- Due to SQLite constraints, FAISS is used as an alternative in-memory vector store.
- FAISS avoids system-level dependencies, allowing embeddings and retrieval to work reliably in the lab environment.

[21]:
```
# 1. Imports
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.prompts import PromptTemplate
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQA

# 2. Load PDF
loader = PyPDFLoader("inputs/Nestle HR Policy.pdf")  # replace with your PDF␣
 ↪path
documents = loader.load()

# 3. Split Text
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000,␣
 ↪chunk_overlap=100)
docs = text_splitter.split_documents(documents)

# 4. Create embeddings and inmemory vector DB
embeddings = OpenAIEmbeddings()
vectordb = FAISS.from_documents(docs, embeddings)

# 5. Prompt template
prompt = PromptTemplate(
    input_variables=["context", "question"],
    template="""You are a helpful HR assistant. Use the context below to answer␣
 ↪the question.
        Context: {context}\n\nQuestion: {question}\nAnswer:"""
)

# 6. Retriever Chain
retriever = vectordb.as_retriever(search_kwargs={"k": 3})
qa = RetrievalQA.from_chain_type(
    llm=ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0),
    chain_type="stuff",
    retriever=retriever,
    chain_type_kwargs={"prompt": prompt}
)
```

[22]:
```
# Gradio Interface
import gradio as gr
print(gr.__version__)
```

```
def answer_question(query):
    return qa.run(query)

iface = gr.Interface(
    fn = answer_question,
     title = "Nestle HR Policy Chatbot",

    inputs = gr.Textbox(label="Ask about HR Policy"),
    outputs = gr.Textbox(label="Bot Answer"),

    flagging_options = ["incorrect response"],
    flagging_dir = "reported_responses" # Reported repsonses will be saved␣
 ↪under this folder in log.csv
)

iface.launch(share=True) # Without share=true, it doesn not load interface␣
 ↪inside the Lab
```

```
4.44.1
Running on local URL:  http://127.0.0.1:7860
Running on public URL: https://28cc5052eeac7c985d.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU
upgrades, run `gradio deploy` from Terminal to deploy to Spaces
(https://huggingface.co/spaces)
```

```
<IPython.core.display.HTML object>
```

[22]:

[20]:
```
#iface.close()
```

```
Closing server running on port: 7860
```

[ ]:
```

```