CDW Documentation

Sample Python Chatbot

Sample Python Chatbot

This requires python with the os, tkinter (tk/ttk), re, time, and openai modules to work.

```
#!/usr/bin/python # replace with path to installed python

import os
import tkinter as tk
from tkinter import ttk, messagebox
import re
import time
from openai import AzureOpenAI, OpenAIError

# === Azure OpenAI Configuration ===
endpoint = "https://<ENDPOINT>.openai.azure.com/"
deployment = "<DEPLOYMENT NAME>"
subscription_key = "<API KEY>"
api_version = "<API_VERSION>" # such as 2024-12-01-preview

client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=subscription_key,
)

# === Initialize conversation memory ===
conversation = [
    {"role": "system", "content": "You are a helpful assistant that is very
sarcastic."} # Change the personality and purpose here.
]

# === PII Pattern Matching ===
PII_PATTERNS = [
    r"\b\d{3}-\d{2}-\d{4}\b",                          # SSN
    r"\b\d{16}\b",                                      # Credit card
(simple)
    r"\b\d{10}\b",                                      # Phone number (10-
digit)
    r"\b[\w\.-]+@[\w\.-]+\.\w+\b",                      # Email
    r"\b\d{3}[-.\s]??\d{3}[-.\s]??\d{4}\b",             # US phone formats
    r"\b(?:\d[ -]*?){13,16}\b",                         # More robust card
pattern
]

def contains_pii(text):
    return any(re.search(pattern, text) for pattern in PII_PATTERNS)

# === Handle sending the user query ===
def send_query():
```

```
    user_input = input_text.get("1.0", tk.END).strip()
    if not user_input:
        messagebox.showwarning("Input Error", "Please enter a message before
sending.")
        return

    if contains_pii(user_input):
        output_text.config(state='normal')
        output_text.insert(tk.END, f"\n⚠ User: {user_input}\n Error: PII
is not allowed in this interface.\n")
        output_text.config(state='disabled')
        input_text.delete("1.0", tk.END)
        return

    try:
        conversation.append({"role": "user", "content": user_input})

        start_time = time.time()
        response = client.chat.completions.create(
            messages=conversation,
            max_completion_tokens=800,
            temperature=1.0,
            top_p=1.0,
            frequency_penalty=0.0,
            presence_penalty=0.0,
            model=deployment
        )
        end_time = time.time()

        latency = end_time - start_time
        reply = response.choices[0].message.content
        conversation.append({"role": "assistant", "content": reply})

        usage = response.usage
        prompt_tokens = usage.prompt_tokens if usage else 0
        completion_tokens = usage.completion_tokens if usage else 0
        total_tokens = usage.total_tokens if usage else prompt_tokens +
completion_tokens

        # Throughput: tokens/sec for this single request
        throughput = total_tokens / latency if latency > 0 else 0

        output_text.config(state='normal')
        output_text.insert(tk.END, f"\n User: {user_input}\n Assistant:
{reply}\n")
        output_text.insert(tk.END, f"\n Stats:\n"
                                   f"  Latency: {latency:.2f} seconds\n"
                                   f"  Prompt Tokens: {prompt_tokens}\n"
                                   f"  Completion Tokens:
{completion_tokens}\n"
                                   f"  Total Tokens: {total_tokens}\n"
```

```
                                    f"  Throughput: {throughput:.2f}
tokens/sec\n")
        output_text.config(state='disabled')
        input_text.delete("1.0", tk.END)

    except OpenAIError as e:
        output_text.config(state='normal')
        output_text.insert(tk.END, f"\n API Error:\nStatus Code:
{getattr(e, 'status_code', 'N/A')}\nMessage: {str(e)}\n")
        output_text.config(state='disabled')
    except Exception as ex:
        output_text.config(state='normal')
        output_text.insert(tk.END, f"\n Unexpected Error:\n{str(ex)}\n")
        output_text.config(state='disabled')

# === GUI Setup ===
root = tk.Tk()
root.title("Azure OpenAI Chat - Sarcastic Assistant")

mainframe = ttk.Frame(root, padding="10")
mainframe.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

# Input Label
ttk.Label(mainframe, text="Enter your message:").grid(row=0, column=0,
sticky=tk.W)

# Input Text Area
input_text = tk.Text(mainframe, width=80, height=4)
input_text.grid(row=1, column=0, sticky=(tk.W, tk.E))

# Send Button
submit_button = ttk.Button(mainframe, text="Send", command=send_query)
submit_button.grid(row=2, column=0, sticky=tk.W, pady=5)

# Output Label
ttk.Label(mainframe, text="Conversation and Metrics:").grid(row=3, column=0,
sticky=tk.W)

# Output Text Area
output_text = tk.Text(mainframe, width=80, height=24, state='disabled',
wrap='word')
output_text.grid(row=4, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))

# Configure resizing
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)
mainframe.columnconfigure(0, weight=1)
mainframe.rowconfigure(4, weight=1)

root.mainloop()
```

AI Knowledge