

***BIT DI PACE (Progetto del gruppo: Cavalli Tommaso, Ouadani Sami, Velluti Mirko, Decolli Andi)***

## **DOCUMENTAZIONE PROGETTO:**

### **Obiettivo del Progetto**

L'obiettivo del progetto è diffondere un linguaggio più "sicuro" in rete. Il programma ascolta l'utente tramite il microfono nella versione DEMO su web, mentre nella versione locale il programma prende l'input da tastiera, trascrive le parole e utilizza un modello di Intelligenza Artificiale in Cloud (tramite l'API di Ollama) per filtrare o riscrivere il contenuto, assicurandosi che il messaggio finale sia educato e pacifico, indipendentemente da come è stato pronunciato originariamente.

#### **1. Configurazione e Inizializzazione**

Il codice su web prepara gli strumenti necessari:

- Sfrutta il motore audio di Chrome o Edge per catturare l'audio dal microfono.
- Configura i parametri audio (frequenza a 16.000 Hz) per garantire che la qualità della registrazione sia compatibile con il riconoscimento vocale.

#### **2. Acquisizione Audio da WEB (Speech-to-Text)**

In questa fase, il programma interagisce con l'hardware:

- Attiva il microfono e rileva quando la persona parla, fermando la registrazione quando essa smette di parlare.
- Cattura le onde sonore e le trasforma in dati digitali (byte).

#### **3. Elaborazione Intelligente (NLP o Natural Language Processing tramite Ollama)**

Questa è la parte decisionale del codice:

- Il testo ottenuto viene unito a un "Prompt di Sistema".
- Queste istruzioni obbligano l'IA a comportarsi come un mediatore: se l'utente è stato aggressivo o offensivo, l'IA deve riscrivere la frase in modo gentile.
- Viene effettuata una chiamata API verso il server Ollama, inviando la frase grezza e ricevendo quella "pacificata".

#### **4. Output dei Risultati**

Infine, il programma mostra a video:

- Frase pacificata in 3 diverse lingue (Italiano,Inglese,Brasiliano): Mostra la versione rielaborata dall'IA, pronta per essere utilizzata in un contesto formale o sicuro.

#### **Requisiti fondamentali:**

VERSIONE LOCALE:

- **Hardware:** un qualsiasi PC.
- **Software:** Python 3.13+
- **Modulo Richiesto,** Requests (installazione in GUIDA ALL'INSTALLAZIONE)

VERSIONE WEB:

- **Hardware:** Pc con connessione alla rete.

#### **Guida all'Installazione per la versione in locale (Setup):**

Dopo aver installato Python 3.13+, nel CMD (WINDOWS) bisogna inserire questa riga di testo:

**pip install requests**

Mentre se si lavora su Linux o MacOs il comando è il seguente:

**pip3 install requests**

**Successivamente bisogna solo avviare il codice fornito.**

**Nota: Per la versione Web bisogna solo premere il tasto “Provalo Sul Browser” sul sito web.**

**Documentazione del Codice: Analisi Riga per Riga**

```
import requests Importa la libreria necessaria per inviare richieste HTTP al server API di Ollama.
```

```
import json Importa la libreria per gestire i dati in formato JSON (formattazione e lettura).
```

```
def ollama_chat_simple(api_key, message, model="cogito-2.1:671b"):
```

Definisce la funzione che comunica con l'IA, accettando la chiave API, il messaggio e il modello.

```
url = "https://ollama.com/api/chat" Specifica l'indirizzo web (endpoint) a cui inviare i dati per la chat.
```

```
headers = { Inizia la creazione dell'intestazione della richiesta HTTP.
```

```
    "Content-Type": "application/json", Indica al server che i dati inviati sono in formato JSON.
```

```
    "Authorization": f"Bearer {api_key}" Inserisce la chiave API per autenticare la sessione di comunicazione.
```

```
} Chiude il dizionario delle intestazioni.
```

```
data = { Inizia la creazione del corpo della richiesta (il "pacchetto" di dati).
```

```
    "model": model, Specifica all'API quale modello di intelligenza artificiale deve rispondere.
```

```
    "messages": [ Inizia la lista che contiene la cronologia o i messaggi della conversazione.
```

```
        {"role": "user", "content": message} Inserisce il messaggio dell'utente identificandolo con il ruolo "user".
```

```
    ], Chiude la lista dei messaggi.
```

```
    "temperature": 0.7, Imposta il livello di creatività dell'IA (0.7 è un valore bilanciato).
```

```
    "max_tokens": 500, Limita la lunghezza massima della risposta a 500 unità di testo (token).
```

```
    "stream": False Chiede al server di inviare la risposta tutta insieme invece che un pezzetto alla volta.
```

```
} Chiude il dizionario dei dati.
```

```
try: Inizia un blocco di codice protetto per gestire eventuali errori di rete o del server.
```

```
    response = requests.post(url, json=data, headers=headers) Invia effettivamente la richiesta POST al server e attende la risposta.
```

```
response.raise_for_status() Verifica se la risposta indica un errore (es. 404 o 500) e nel caso solleva un'eccezione.
```

```
result = response.json() Converte la risposta grezza del server in un oggetto Python (dizionario).
```

```
if 'message' in result: Controlla se nella risposta del server è presente la chiave 'message'.
```

```
return result['message']['content'] Estraе e restituisce solo il testo della risposta contenuto in 'message'.
```

```
elif 'response' in result: Verifica se il testo è presente sotto la chiave alternativa 'response'.
```

```
return result['response'] Restituisce il testo trovato nella chiave 'response'.
```

```
else: Viene eseguito se il formato della risposta non è tra quelli previsti.
```

```
return f"Formato risposta non riconosciuto: {json.dumps(result, indent=2)[:200]}..." Restituisce una parte della risposta ricevuta per aiutare il debug.
```

```
except requests.exceptions.HTTPError as e: Cattura errori specifici legati alla comunicazione web (es. sito non raggiungibile).
```

```
return f"Errore HTTP: {e.response.status_code} - {e.response.text}" Restituisce i dettagli dell'errore HTTP avvenuto.
```

```
except Exception as e: Cattura qualsiasi altro tipo di errore generico del programma.
```

```
return f"Errore: {str(e)}" Restituisce la descrizione dell'errore generico.
```

```
if __name__ == "__main__": Assicura che il codice sottostante venga eseguito solo se il file viene lanciato direttamente.
```

```
API_KEY =
"4b343ffcd5f34c2f9e4366f80bc33f74.HVAAdJLUzPjH2tVX0s9vsAfk" Assegna la chiave di accesso personale per le API di Ollama.
```

```
print("Scegli la lingua / Choose language / Escolha o idioma:")
Mostra a video il titolo del menu di selezione lingua.
```

```
print("1. Italiano (ITA)") Mostra l'opzione per la lingua italiana.
```

```
print("2. English (ENG)") Mostra l'opzione per la lingua inglese.
```

```
print("3. Português Brasileiro (PT-BR)") Mostra l'opzione per la lingua portoghese.
```

```
scelta = input("Scelta (1/2/3): ").strip() Legge l'input dell'utente e rimuove eventuali spazi bianchi superflui.

if scelta == "2": Inizia il blocco di configurazione se l'utente ha scelto l'Inglese.

stringHeader = "I am passing you a phrase that might contain something offensive/aggressive..." Imposta le istruzioni per l'IA (prompt) in lingua inglese.

msg_welcome = "💬 Simple chat with Ollama Cloud" Imposta il messaggio di benvenuto in inglese.

msg_exit = "Type 'exit' to quit\n" Imposta il comando di chiusura in inglese.

msg_result = "peaceful phrase" Imposta l'etichetta del risultato in inglese.

elif scelta == "3": Inizia il blocco di configurazione se l'utente ha scelto il Portoghese.

stringHeader = "Vou te passar uma frase que pode conter algo ofensivo/agressivo..." Imposta le istruzioni per l'IA (prompt) in lingua portoghese.

msg_welcome = "💬 Chat simples com Ollama Cloud" Imposta il messaggio di benvenuto in portoghese.

msg_exit = "Digite 'exit' para sair\n" Imposta il comando di chiusura in portoghese.

msg_result = "frase pacificada" Imposta l'etichetta del risultato in portoghese.

else: Blocco predefinito per la scelta "1" o qualsiasi altro input (Italiano).

stringHeader = "ti passo una frase che potrebbe contenere qualcosa di offensivo/aggressivo..." Imposta le istruzioni per l'IA (prompt) in lingua italiana.

msg_welcome = "💬 Chat semplice con Ollama Cloud" Imposta il messaggio di benvenuto in italiano.

msg_exit = "Scrivi 'exit' per uscire\n" Imposta il comando di chiusura in italiano.

msg_result = "frase pacificata" Imposta l'etichetta del risultato in italiano.

print(f"\n{msg_welcome}") Stampa a video il messaggio di benvenuto scelto.

print(msg_exit) Stampa a video le istruzioni su come uscire dal programma.

while True: Inizia un ciclo infinito per permettere una chat continua.
```

```
raw_input = input("User: ") Attende l'inserimento di una frase da parte dell'utente.  
  
if raw_input.lower() == 'exit': Controlla se l'utente ha digitato "exit" (in qualsiasi formato maiuscolo/minuscolo).  
  
break Esce dal ciclo infinito e chiude il programma.  
  
user_payload = stringHeader + raw_input Unisce le istruzioni della lingua scelta alla frase scritta dall'utente.  
  
response = ollama_chat_simple(API_KEY, user_payload) Chiama la funzione definita all'inizio per ottenere la risposta dall'IA.  
  
print(f"\n{msg_result}: {response}\n") Stampa la frase elaborata dall'IA con l'etichetta della lingua corretta.
```

## Documentazione Terminata: Grazie per la Lettura

©2026 - Bit di Pace