**DATA SET**

PolyGlot bot does not rely on a fixed, pre-existing dataset. Instead, it dynamically generates responses and translations using real-time AI models and APIs. The bot operates on, pre-existing dataset, which sets it apart from traditional rule-based or static-response systems. Instead of pulling responses from a predefined database of answers, it leverages powerful, real-time AI models and APIs such as OpenAI’s language models or translation services to generate responses dynamically. This means that when a user inputs a question or a statement, the bot processes the input on the spot, understands the context, and creates a unique, context-aware response in the desired language. This approach allows the PolyGlot bot to be highly flexible, adaptive, and capable of handling a wide range of topics and languages with natural fluency. Additionally, by integrating multilingual capabilities, it can translate and respond in various languages seamlessly, making it ideal for global communication, education, and customer support scenarios. The data sources include:

**1. Google Gemini AI (for Natural Language Processing & Responses)**

The Gemini AI model processes user queries, understands intent, and generates meaningful responses based on its vast pre-trained transformer architecture. It does not depend on a static dataset but draws on a rich knowledge base learned from diverse sources, including books, websites, and scholarly papers. This allows PolyGlot to provide accurate, conversational replies without needing manual training or scripting.

# Set up Google Gemini API Key

genai\_configure(api\_key="AIzaSyAESbu-jTTS3j9rR\_Yo58WMZmdhE4Yz8j0")

# Preload Gemini AI model for better performance

model = genai.GenerativeModel("gemini-1.5-pro")

def get\_ai\_response(user\_input):

    try:

        response = model.generate\_content(user\_input)

        return response.text

    except Exception as e:

        return "⚠️ Error: Unable to process request."

**2. Google Translate API (for Language Translation)**

For multilingual interaction, the bot utilizes the Google Translate API, which handles language translation between English and several Indian languages. This API is powered by machine learning models trained on large multilingual corpora such as bilingual texts, online articles, and parallel translation datasets. The use of this API ensures accurate, contextual translation that reflects real-world language usage.

# Initialize the translator

translator = Translator()

# Function to generate and encode audio with correct pronunciation

def translate\_text(text, target\_lang="en"):

    try:

        translated = translator.translate(text, dest=target\_lang)

        return translated.text

    except Exception as e:

        return "⚠️ Error: Translation failed."

**3. GTTS (Google Text-to-Speech) API (for Audio Output Generation)**

PolyGlot uses the GTTS library to generate spoken audio from text. This service connects to Google’s advanced text-to-speech engine, which uses deep learning models trained on extensive voice data from a wide range of speakers and accents. As a result, it can produce clear, natural-sounding speech in multiple languages, all without requiring any separate voice dataset for your application.

def generate\_audio\_base64(text, lang="en"):

    try:

       sanitized\_text = clean\_text(text)  # Clean text for better speech output

        tts = gTTS(text=sanitized\_text, lang=lang, slow=False)  # Ensure proper pronunciation

        audio\_path = "response\_audio.mp3"

        tts.save(audio\_path)

        # Convert audio to base64

with open(audio\_path, "rb") as audio\_file:

            audio\_base64 = base64.b64encode(audio\_file.read()).decode("utf-8")

# Custom-styled audio player inside chat box

        return f"""

        <audio controls style="width:100%; margin-top:5px; border-radius:5px; background:#ddd; height:30px;">

            <source src='data:audio/mp3;base64,{audio\_base64}' type='audio/mp3'>

        </audio>

        """

    except Exception as e:

        return "⚠️ Error: Unable to generate audio."

**4. User-Generated Data (Real-Time Input from Users)**

Every input processed by PolyGlot comes directly from the user in real time. It does not store previous interactions or learn from user history. Instead, it processes each message independently and dynamically, generating relevant responses on-the-fly through inference, which makes the chatbot highly flexible and responsive without being tied to a fixed dataset. PolyGlot bot relies heavily on user-generated data as a core part of its real-time interaction process. Instead of using static scripts or pre-programmed dialogues, the bot takes live input directly from users—questions, commands, or messages—and processes it instantly. This real-time data becomes the foundation for generating responses or performing translations. By analyzing the user’s input dynamically, the bot can tailor its replies to suit the specific context, intent, and language preferences of each individual interaction. This user-driven approach not only enhances personalization but also ensures that the bot remains relevant and responsive in dynamic conversation flows.

**Why PolyGlot Doesn’t Need a Fixed Dataset?**

Unlike traditional rule-based or FAQ-driven bots, which rely on hardcoded datasets, PolyGlot operates entirely on real-time AI inference using cloud-based language models. This gives it the capability to understand a wide variety of inputs, adapt to different languages and contexts, and provide up-to-date responses without requiring a separate dataset for each language or topic. Its multilingual capability is not limited by static corpora, enabling seamless switching and handling of diverse linguistic input.

**Pretrained Dataset for Custom User Responses:**

There is a small dictionary of preprogrammed replies under pretrained\_qna, which covers basic greetings and common queries like “hi,” “hello,” “how are you,” or “what is your name.” These are answered instantly without invoking the Gemini model, to speed up simple conversations. For any input outside this dictionary, the bot forwards the message to the Google Gemini AI model, which handles the query dynamically and generates a response based on its large-scale training data.

pretrained\_qna = {

"hi": "Hello! How can I help you today?",

"hello": "Hi there! What would you like to talk about?",

"how are you": "I'm just a bot, but I'm functioning at 100%!",

"what is your name": "I'm PolyGlot, your multilingual AI assistant.",

"help": "Sure! You can ask me questions in multiple Indian languages, and I’ll respond with text and voice."

}

For everything else not in the above dictionary, the input is sent to the Google Gemini model, which generates a reply on-the-fly using its own training data

**Libraries to import use all the preloaded datasets:**

import google.generativeai as genai

import ipywidgets as widgets

from IPython.display import display, HTML, Javascript

from gtts import gTTS

from googletrans import Translator

import base64

import re  # Added for text sanitization

**Future Scope – Adding a Custom Dataset:**

In future if we want PolyGlot to be fine-tuned for specific domains (e.g., medical, legal, or technical support), we can:

* Train a custom NLP model using Hugging Face datasets or TensorFlow/Keras.
* Use Lang-Chain or RAG (Retrieval-Augmented Generation) to feed domain-specific data into Gemini AI.
* Integrate speech datasets from Mozilla Common Voice or Open-SLR for improved voice generation.

However, for general-purpose use, PolyGlot is already highly efficient as it leverages the power of Google’s advanced AI and translation models.