**AI Voice Chat-bot Dataset Documentation**

**Overview**

This document provides guidelines on how to structure, feed, and train the AI voice chat-bot using a dataset. The chat-bot utilizes a dataset-based approach to classify user intents and generate appropriate responses.

**1. Dataset Structure**

The dataset should be structured as a list of dictionaries, where each dictionary represents a conversation sample. Each entry should include:

* **user\_query** (str): The sample user input.
* **intent** (str): The intent label assigned to the query.
* **bot\_response** (str): The chat-bot's response for the given intent.

**Example Dataset Entry:**

dataset = [

{"user\_query": "What are your services?", "intent": "services\_info", "bot\_response": "We offer web design, marketing, and branding services."},

{"user\_query": "How can I contact you?", "intent": "contact\_info", "bot\_response": "You can contact us via email at support@iconicdreamfocus.com."}

]

**2. Feeding Data into the Model**

The dataset is used to train an intent classification model. Follow these steps to process the dataset:

1. Extract user queries and intent labels:
2. user\_queries = [item["user\_query"] for item in dataset]

intents = [item["intent"] for item in dataset]

1. Convert text data into numerical vectors using TF-IDF:
2. from sklearn.feature\_extraction.text import TfidfVectorizer
3. vectorizer = TfidfVectorizer()

X = vectorizer.fit\_transform(user\_queries)

1. Train the intent classification model:
2. from sklearn.linear\_model import LogisticRegression
3. intent\_model = LogisticRegression(random\_state=0)

intent\_model.fit(X, intents)

**3. Training the AI Chat-bot**

**Step 1: Preparing the Model**

Ensure that the dataset is correctly formatted before training. Expand the dataset by adding more variations of user queries for each intent to improve accuracy.

**Step 2: Training Execution**

Run the following code snippet to train the chat-bot:

intent\_model.fit(X, intents)

**Step 3: Validating the Model**

After training, test the model using unseen queries:

user\_input = "Tell me about your services"

input\_vector = vectorizer.transform([user\_input])

predicted\_intent = intent\_model.predict(input\_vector)[0]

print(f"Predicted Intent: {predicted\_intent}")

**4. Expanding and Updating the Dataset**

To improve chat-bot performance:

* Regularly update the dataset with new user queries.
* Include multiple variations for each intent.
* Fine-tune responses for better engagement.
* Retrain the model whenever significant dataset changes occur.

**5. Deployment Considerations**

* Load the trained model once to avoid redundant processing.
* Use caching (st.cache\_resource) in Streamlit for efficiency.
* Ensure that the chat-bot supports both text and voice inputs for better accessibility.

**6. Future Improvements**

* Implement a more advanced deep learning-based model (e.g., transformers or RNNs) for intent classification.
* Use an external database for managing and updating chat-bot responses dynamically.
* Enhance speech recognition with real-time Whisper API integration.