# Unity to Serve Chatbot - README

Welcome to the **Unity to Serve Chatbot**! This chatbot is designed to provide intelligent, user-friendly, and mission-driven answers about the Unity to Serve organization. It combines rule-based logic, semantic similarity, and fuzzy matching to handle a wide range of queries.

## 📌 Purpose

The chatbot offers fast, accurate, and helpful responses about Unity to Serve’s mission, leadership, donation options, programs, and values. It’s built to:

* Guide users through common inquiries.
* Offer fallback suggestions when uncertain.
* Log unmatched inputs for future improvement.

## 🧠 Features

| Feature | Description |
| --- | --- |
| **Rule-based Responses** | Uses a structured JSON FAQ to match known intents and return predefined answers. |
| **spaCy NLU** | Detects user intent via spaCy-based classification. |
| **Semantic Search** | Uses Sentence Transformers to find the closest match when spaCy is unsure. |
| **Fuzzy Matching** | Applies token-based fuzzy logic for typo-prone or loosely phrased queries. |
| **Fallback Suggestions** | Suggests 3 closest alternatives if no good match is found. |
| **Input Logging** | Logs unknown inputs and suggestions to a JSON file for retraining. |

## 🛠️ Project Structure

.  
├── main.py # FastAPI app entry point  
├── nlp\_handler\_phase3.py # spaCy NLU intent classifier  
├── rule\_responses\_phase3.py # faq\_dict + get\_response() function  
├── semantic\_handler.py # Semantic search logic (SentenceTransformer)  
├── fuzzy\_handler.py # Fuzzy matching logic (FuzzyWuzzy)  
├── logger\_phase3.py # Logs unmatched queries  
├── unmatched\_inputs.json # Stored logs of unmatched inputs  
├── unity\_faq\_dict.json # Core FAQ database with intents and answers  
└── README.md # 📄 You're here!

## 🔄 Chatbot Workflow

User input  
 ↳ spaCy intent + confidence  
 └── ✓ If confidence >= 0.65 ➔ rule-based reply  
 └── ✗ Else ➔ check semantic similarity (threshold: 0.65)  
 └── ✓ Semantic response if match  
 └── ✗ Else ➔ check fuzzy match (score >= 0.65)  
 └── ✓ Fuzzy response if match  
 └── ✗ Else ➔ show fallback suggestions (top 3)

## ⚙️ Initialization

During FastAPI app startup:

model = SentenceTransformer('all-MiniLM-L6-v2')  
faq\_embeddings = model.encode(list(faq\_dict.keys()), convert\_to\_tensor=True)

This enables semantic similarity via cosine score matching.

## 📂 unity\_faq\_dict.json Example

{  
 "mission\_overview": "Unity to Serve is dedicated to fostering sustainable...",  
 "donate\_inquiry": "You can donate securely at https://unitytoserve.org/donate",  
 "who is Mohammad Ramin Khanzada?": "Khanzada Mohammad Ramin is the founder..."  
}

## 🔍 Fallback Example

**User Input**: “What do you guys do?”

[DEBUG] spaCy Intent: mission\_overview, Confidence: 0.30  
[SEMANTIC DEBUG] Best match: 'greeting' with score: 0.3229  
[DEBUG] Fuzzy Match: None, Score: 0.00  
[DEBUG] Fallback Suggestions: ['greeting', 'contact\_us', 'leadership and board']

**Bot**:

I'm still learning and working to improve my answers.  
Did you mean one of the following?  
- greeting  
- contact\_us  
- leadership and board

## 📝 Logging Unknown Inputs

Unmatched queries are stored like this:

{  
 "timestamp": "2025-07-30T13:15:42",  
 "user\_input": "whats ur logo mean?",  
 "chatbot\_response": "- what does the Unity to Serve logo represent?..."  
}

## 🔬 Test Cases (Sample)

| Test Input | Expected Result |
| --- | --- |
| “How can I donate to Unity to Serve?” | Direct rule-based answer with donate link |
| “wht are the core valuess” | Semantic or fuzzy match to core values response |
| “asdfasljf123” | Fallback message with top 3 suggestions |
| “who is the foundr of unity to srv” | Fuzzy match to founder answer |

## 🧠 Is This Chatbot Smart?

Yes! It’s smart within its designed boundaries:

* Understands various phrasings.
* Gives fallback help.
* Avoids hallucinating.
* Continuously improves via logging.

## 📢 Chatbot Decision Matrix

| Stage | Trigger | Expected Outcome |
| --- | --- | --- |
| ✅ Step 1 | spaCy intent match ≥ 0.65 | Returns rule-based response |
| ✅ Step 2 | Semantic match ≥ 0.65 | Returns rule-based response |
| ✅ Step 3 | Fuzzy match ≥ 65 | Returns rule-based response |
| ✅ Step 4 | No matches ≥ 0.65 | Returns fallback with top 3 suggestions ≥ 0.1 |
| ✅ Step 5 | All failed | Logs input + returns fallback message |

### Libraries

| Library | Role |
| --- | --- |
| **spaCy** | ✅ Primary check |
| **SentenceTransformer** | ✅ Meaning-based match |
| **FuzzyWuzzy** | ✅ Typo-resistant match |
| **Suggestion Deduping** | ✅ Clean output |
| **Logger** | ✅ Learning-ready |

### Step-by-Step Explanation

1. **Get intent and confidence using spaCy**  
   • Example: “about us” → intent: about, confidence: 0.82
2. **If confidence ≥ 0.65, return response from** ``  
   • ✅ If found → return it
3. **If intent fails or confidence < 0.65, use semantic search**  
   • Compare input to all keys using SentenceTransformer • If cosine similarity ≥ 0.65 → return match
4. **If semantic fails, apply fuzzy match**  
   • Use FuzzyWuzzy to compare user input to keys • If score ≥ 65 → return match
5. **If nothing matches, show top 3 suggestions with similarity ≥ 0.1** • Suggestions are NOT used for actual matching
6. **Log unmatched input + suggestions** • Logs are saved to unmatched\_inputs.json
7. **Return fallback message**

I'm still learning and working to improve my answers.  
Did you mean one of the following?  
- About Us  
- Our Mission  
- Donate Now

## 🔬 Common AI Terms (Simplified)

| Term | What It Is | Role in AI |
| --- | --- | --- |
| **AI** | A broad field about making machines intelligent | Big umbrella |
| **ML** | Subset of AI where models learn from data | Inside AI |
| **Deep Learning** | Subset of ML using neural networks | Inside ML |
| **Transformer** | Deep learning model architecture | Inside DL |
| **GPT, BERT** | Pretrained transformer models | Real-world AI systems |

## 💼 Library Examples in Practice

* **spaCy**: Like checking a dictionary to instantly recognize what someone means by “donate”.
* **SentenceTransformer**: Like understanding that “How can I give money?” is similar to “donate” even if the word isn’t used.
* **FuzzyWuzzy**: Helps if someone types “donat” instead of “donate” — it still figures it out.
* **Logging**: Records what the bot couldn’t answer so we can teach it later.

## 👥 Maintainers

Developed by Mustafa Waizy for the Unity to Serve project.

For issues or contributions, contact: [unitytoserve.org](https://unitytoserve.org)

Thank you for using the Unity to Serve Chatbot!