We'll cover **every stage**:

| **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**

|  |  |
| --- | --- |
| **spaCy intent match** | **✅ Primary check** |
| **SentenceTransformer** | **✅ Meaning-based match** |
| **FuzzyWuzzy fallback** | **✅ Typo-resistant match** |
| **Deduped suggestions** | **✅ Clean output** |
| **Logging unmatched input** | **✅ Learning-ready** |

**What Happens — Step-by-Step Explanation**

1. **Get intent and confidence using spaCy**
   * Example: "about us" → intent: about, confidence: 0.82
2. **If confidence ≥ 0.65**, check if there's a matching response in faq\_dict
   * ✅ If found → return it
3. **If intent fails or confidence < 0.65**, move to **semantic similarity**
   * Compare input against all FAQ keys using SentenceTransformer
   * If cosine similarity ≥ 0.65 → return matching response
4. **If semantic similarity also fails**, move to **fuzzy string match**
   * Use fuzzywuzzy to find the best-matching key
   * If score ≥ 65% → return response
5. **If none of the above work**, get top 3 suggestions where similarity ≥ 0.1
   * These are just guesses; used for display only, not actual response matching
6. **Log** the user input + suggestions for later training
   * Log stored in unmatched\_inputs.json
7. **Return fallback message**

**text**

**CopyEdit**

**I'm still learning and working to improve my answers.**

**Did you mean one of the following?**

**- About Us**

**- Our Mission**

**- Donate Now**

| **Term** | **What It Is** | **Role in AI** |
| --- | --- | --- |
| **AI** | A broad field about making machines intelligent | Big umbrella |
| **Machine Learning (ML)** | A subset of AI where models learn from data | Inside AI |
| **Deep Learning** | A subset of ML using neural networks | Inside ML |
| **Transformer** | A specific deep learning architecture (model structure) | Inside DL |
| **GPT, BERT, T5, etc.** | Pretrained models built using transformers | Real-world AI systems |

**✅ Best Real-World Approach: Hybrid Human-in-the-Loop Self-Learning**

**🔄 1. Log Unknown Inputs Automatically *(✔ Must-have)***

Every good chatbot logs:

* What it couldn't answer
* When it happened
* What fallback suggestions were shown

🛠 Already covered in your case with unmatched\_inputs.json.

**🧠 2. Semi-Automated Suggestions *(✔ Real-world smart upgrade)***

Don't jump straight to auto-learning. Instead:

**✅ Best Practice:**

* Use SentenceTransformers to compute similarity between unknown input and existing FAQs.
* If similarity > 0.75, suggest a matching intent or FAQ.
* Let **you** or an **admin** approve it.

You can even build a simple:

* admin\_review.py CLI script
* or a Streamlit dashboard  
  to approve/reject suggestions.

This keeps quality high **without full retraining risk**.

**👨‍🏫 3. Human Curation Weekly *(✔ Real orgs do this)***

Organizations like customer service teams or nonprofits:

* Export unmatched\_inputs.json weekly
* Review entries
* Add new entries to:
  + faq\_dict.json
  + training\_examples.json (for spaCy)
* Retrain if needed

✅ Easy, safe, no broken logic.

**❌ What Doesn't Work Well in Reality**

| **Approach** | **Why it fails** |
| --- | --- |
| ❌ Full Auto-Learning | Risk of adding wrong answers or low-quality ones |
| ❌ Relying only on spaCy/PhraseMatcher | Misses meaning-based (semantic) intent |
| ❌ Ignoring feedback | Bot never improves, frustrates users |