VITYARTHI PROJECT

Python Chatbot

By:

Gahira Kaur

(25BAI10026)

Dr. Antima Jain

Faculty

VIT®

BHOPAL

www.vitbhopal.ac.in

VIT Bhopal University

Kotrikalan – 466114

Madhya Pradesh,

INDIA.

November 2025

1. Introduction

The objective of this project is to create a Chabot in python that responds to the user's questions. The code works on the principle of probability, calculating a percentage based on how many keywords actually match the question input by user providing much better matchibility to the correct response to the user's questions.

2. Problem Statement

Colleges often get many questions regarding location, admissions, placements, and specific contact details from aspirants. To handle such basic questions, we can use the help of a Chabot rather than hiring someone to answer these basic questions about the institute. The problem is to automate the handling of these common queries using a robust, text-based interface to improve efficiency and provide 24/7 availability of information.

3. Functional Requirements

- **Intent Recognition:** The system must accurately identify what the user desires to know based on their question.
- **Keyword Matching:** The system must match the words in the question against a list of predefined keywords associated with each response.
- **Required Word Enforcement:** The feature of using required words enforces the strength of keyword matching and makes sure that the wrong response is not given to a question.
- **Response Generation:** Upon a successful match, the system must return a specific response.
- **Handling Unknown Input:** If the Chabot receives a question for which a response has not been programmed, the Chabot must have a fallback statement to provide.
- **Case and Punctuation Agnostic:** The system must process user input irrespective of capitalization or punctuation.

4. Non-Functional Requirements

- **Performance:** The response time must be very small due to the simplicity of the algorithm.
- **Maintainability:** The keywords and responses must be easy to update and modify.
- **Reliability:** The core logic must ensure a high probability of finding the correct response for known queries.

5. System Architecture

The Chabot is created in two separate files. The files are separated to simplify the code and make it easier to understand.

python_chatbot_project: Contains the main functions for input processing, probability calculation, intent checking, and the main conversational loop.

long_respnses: Stores all the predefined responses and the random fallback replies.

6. Design Diagram

Workflow Diagram

The user's input flows through the following path:

User input → check_all_messages()→response()→message_probability()→highest score selection →Output

7. Design Decisions & Rationale

A. Intent Scoring (`message_probability` function)

Design Rationale: A simple percentage-based score is calculated to measure the match strength.

Score = (number of recognized words/total number of words in the question) x 100

This ensures that if a user uses more words relevant to the intent, the score is higher. The percentage is converted to an integer for simplicity.

B. Required Word Enforcement

Design Rationale: To ensure better keyword matching, the `required_words` list is created.

Since a lot of questions may use the same filler words, creating a required words list for each response ensures better matchibility of the appropriate response.

C. Input Pre-processing (`get_response` function)

Design Rationale: To achieve case and punctuation independence, the input is altered:

1. It is converted to lowercase using `.lower() function`.
2. It is split into a list of words using a regular expression `re.split` that handles spaces and common punctuation marks (`,;?!.-`) as delimiters. This ensures that words like "hello!" are correctly reduced to "hello".

8. Implementation Details

A. `long_responses.py` Structure

This secondary file consists of all the responses to the questions defined in the main file. In the main file all the keywords are assigned to separate functions that get sent to this file. The responses are stored here

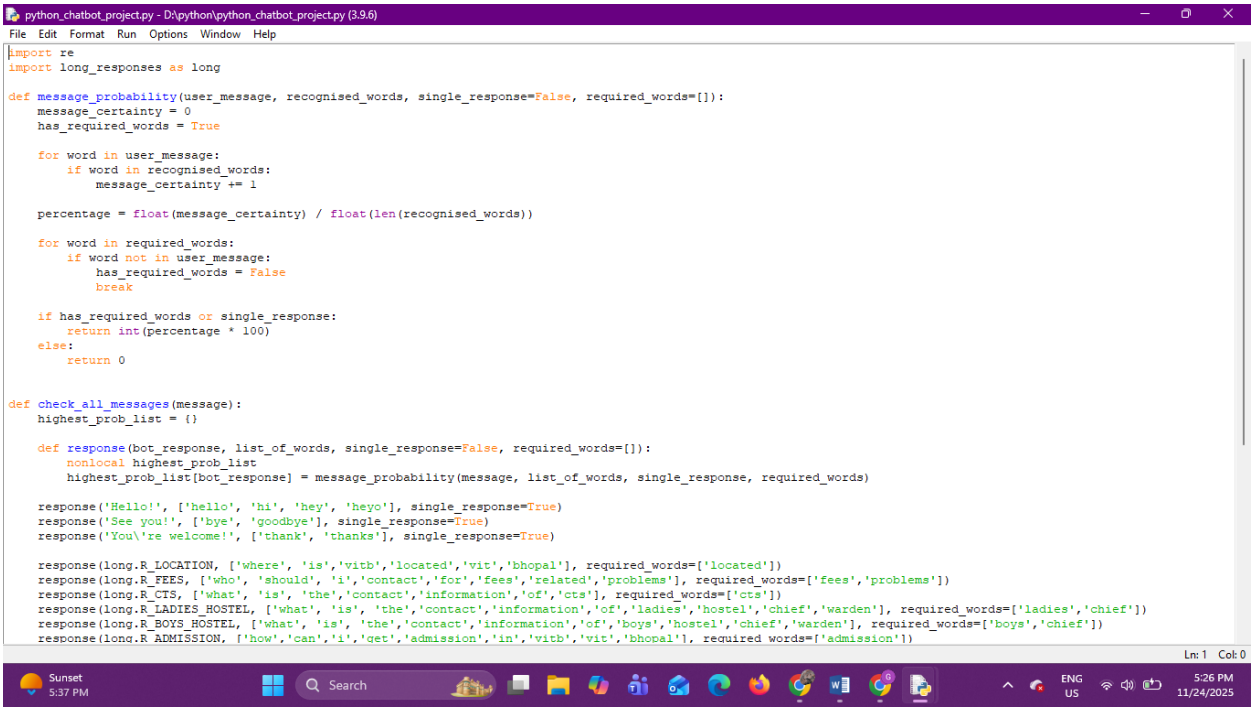
B. Intent Definition

The `check_all_messages` function centralizes the scoring process. It uses a nested function `response` to simplify the definition of intents. Each intent is defined by:

1. The **response text**.
2. The **list of recognized words**.
3. The **`single_response`** flag.
4. The **`required_words`** list.

The functionchecks all the defined intents, calculates a score for each, and uses `max(highest_prob_list, key=highest_prob_list.get)` to select the response with the maximum score. A threshold of < 1 is used to trigger the `long.unknown()` response.

9.Screenshots



```
    response(long.R_ADMISSION, ['how','can','i','get','admission','in','vitb','vit','bhopal'], required_words=['admission'])
    response(long.R_PLACEMENTS, ['what', 'are','the','placements','statistics','of','vitb','vit','bhopal'], required_words=['placements'])
    response(long.R_CHANCELLOR, ['who','is','the','chancellor','of','vitb','vit','bhopal'], required_words=['chancellor'])
    response(long.R_AGE, ['how','old','is','vitb','vit','bhopal'], required_words=['old'])


    best_match = max(highest_prob_list, key=highest_prob_list.get)

    return long.unknown() if highest_prob_list[best_match] < 1 else best_match


def get_response(user_input):
    split_message = re.split(r'\s+|[,;?!.-]\s*', user_input.lower())
    response = check_all_messages(split_message)
    return response


while True:
    print('Bot: ' + get_response(input('You: ')))
```

long_responses.py - D:\python\long_responses.py (3.9.6)

File Edit Format Run Options Window Help

```
import random

R_LOCATION = "VIT Bhopal is located in Kotri Kalan, Astha, Sehore"
R_FEES = "Contact this email id for your fees related problems- onlinefees@ismoman.com "
R_CTS = "Contact this email id for cts office- cts@vitbhopal.ac.in"
R_LADIES_HOSTEL = "This is the email id of girls hostel chief warden- cw.lh@vitbhopal.ac.in"
R_BOYS_HOSTEL = "This is the email id of boys hstel chief warden- cw@vitbhopal.ac.in"
R_ADMISSION = "For admission into vitb you must clear VITEEE"
R_PLACEMENTS = "90% if students from vitb get placed"
R_CHANCELLOR = "The chancellor of vitb is Dr. G. Viswanathan"
R_AGE = "Vitb is 10 years old"
def unknown():
    response = ["Could you please re-phrase that? ",
                "I'm sorry, I don't understand",
                "I couldn't quite process that",
                "What do you mean by that?"][
        random.randrange(4)]
    return response
```

Output

IDLE Shell 3.9.6

File Edit Shell Debug Options Window Help

```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\python\python_chatbot_project.py ==================
You: hello
Bot: Hello!
You: where is vitb located
Bot: VIT Bhopal is located in Kotri Kalan, Astha, Sehore
You: contact information of ladies hostel chief warden
Bot: This is the email id of girls hostel chief warden- cw.lh@vitbhopal.ac.in
You: hjkhkl
Bot: I couldn't quite process that
You: bye
Bot: See you!
You:
```

10. Testing Approach

The code was tested by simply running the code and inputing the questions coded along with some questions that were not coded to test the fallback function.

We run many iterations of the code and use different inputs to ensure that the code works properly.

11. Challenges Faced

The challenges I faced in the making of this project were:

- Coming up with a unique idea.
- Understanding the use of the python libraries that have been used in the project.
- Debugging process.

12.Learnings and key takeaways

This project helped me learn a lot about coding in Python. It helped me understand how logic flows in programming and how to design and implement a code based in real world applications.

13. Future Enhancements

While functional, the current system has limitations due to its rule-based nature. Future enhancements could include:

1. **Stemming/Lemmatization:** Introduce a pre-processing step to reduce words to their base form to expand keyword coverage without manually listing all variants.
2. **Jaccard Similarity:** Replace the current simple percentage score with a more mathematically robust measure like the Jaccard index to better handle long user inputs with few keywords.
3. **Database Integration:** Migrate the static responses from a Python file (`long_responses.py`) to an external database to allow the content to be updated without modifying the core code.
4. **Context Management:** Implement basic state or session tracking to handle simple follow-up questions.

14. References

1. Python