CREATE A CHATBOT IN PYTHON

SUBMITTED BY

SHANMUGAPPRIYK(311521106089)

SUBIDSHA S C(311521106100)

SWARNALAKSHMI E(311521106102)

SOUNDHARYA B(311521106094)

PHASE 2 SUBMISSION :INNOVATION

Creating a chatbot in Python involves several steps. Here's a high-level overview of the process:

1. **Define Your Chatbot's Purpose**:

Determine the purpose and scope of your chatbot. Is it for customer support, entertainment, information retrieval, or something else? Understanding the chatbot's goal will guide your development process.

2. **Choose a Chatbot Type**:

Decide whether you want to create a rule-based chatbot or a machine learning-based chatbot. Rule-based chatbots follow predefined rules, while machine learning-based chatbots can learn from data.

3. **Gather Data** (for Machine Learning-Based Chatbots):

If you're building a machine learning-based chatbot, you'll need a dataset of questions and answers. You can create your dataset or use an existing one.

4. **Select a Framework or Library**:

Choose a framework or library for building your chatbot. Common choices include:

- For rule-based chatbots: Python's NLTK (Natural Language Toolkit) or spaCy.

- For machine learning-based chatbots: TensorFlow, PyTorch, or specialized libraries like Rasa NLU or Dialogflow.

5. **Data Preprocessing**:

If you're working with a machine learning-based chatbot, you'll need to preprocess your data. This may include text tokenization, removing stop words, and data cleaning.

6. **Design Your Chatbot's Conversational Flow**:

Create a flowchart or plan for how your chatbot will interact with users. Define possible user inputs and how your chatbot will respond.

7. **Develop the Chatbot**:

Write the code for your chatbot. This includes defining functions to handle user inputs, process data, and generate responses.

8. **Train Your Chatbot** (for Machine Learning-Based Chatbots):

If you're using machine learning, train your chatbot model on your dataset. This involves defining and training the model architecture.

9. **Test Your Chatbot**:

Test your chatbot with sample inputs to ensure it understands and responds correctly. Make adjustments as needed.

10. **Integrate NLP (Natural Language Processing)**:

Implement Natural Language Processing techniques to improve the chatbot's understanding of user input. This may include entity recognition, sentiment analysis, and intent classification.

11. **Handle Special Cases**:

Account for scenarios where the chatbot encounters unexpected or out-of-scope questions.

12. **Deploy Your Chatbot**:

Deploy your chatbot to a platform where users can interact with it. This could be a website, a messaging app, or a custom application.

13. **Monitor and Maintain**:

Continuously monitor your chatbot's performance and gather user feedback. Make updates and improvements based on user interactions.

14. **Scale and Optimize**:

As your chatbot gains more users, ensure it can scale to handle increased demand. Optimize its performance and response times.

15. **Privacy and Security**:

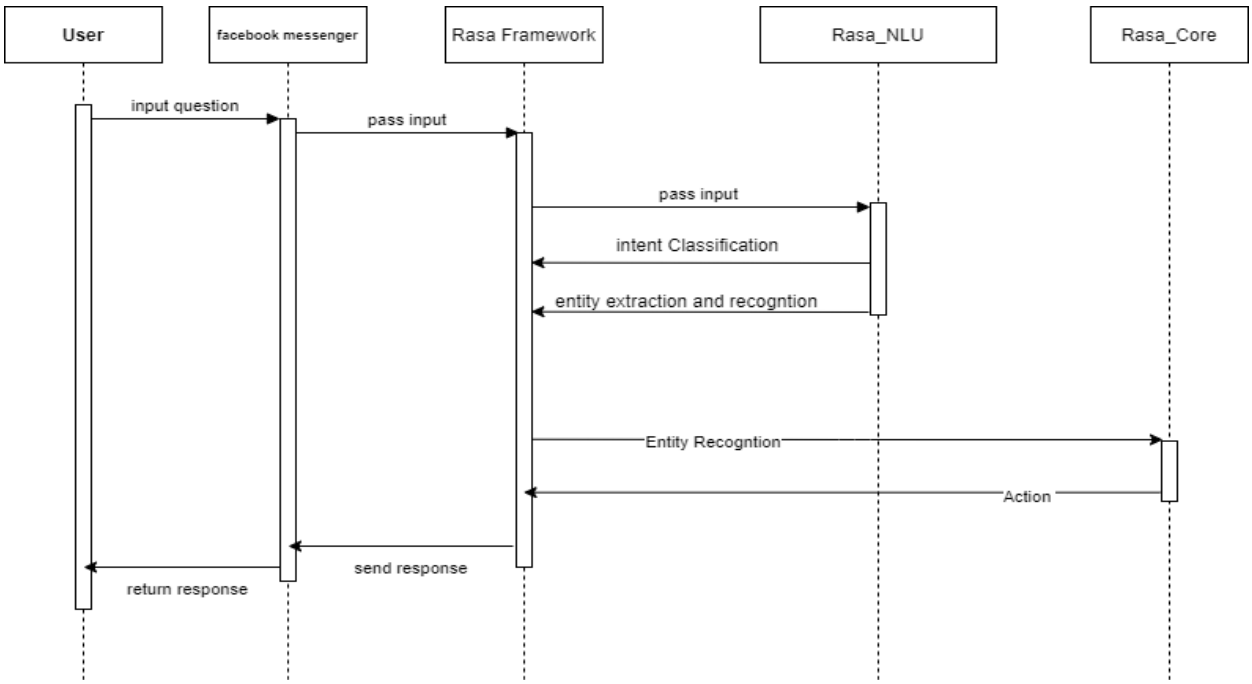
If your chatbot handles sensitive data, consider implementing privacy and security measures to protect user information.

16. **Documentation and User Training**:

Provide documentation and training materials for users to understand how to interact with your chatbot effectively.

CREATING A DATASET FOR CHATBOT IN PYTHON:

Creating a dataset for training a chatbot involves collecting pairs of questions and answers. You can create your own dataset or use existing chatbot datasets. One popular dataset for



chatbot training is the Cornell Movie Dialogs Corpus, which contains a collection of movie character dialogues. Here's how you can obtain and use it in Python:

1. **Download the Dataset**:

   You can download the Cornell Movie Dialogs Corpus from this link: [Cornell Movie Dialogs Corpus](https://www.cs.cornell.edu/~cristian/Cornell_Movie_Dialogs_Corpus.html). Download the ZIP file and extract its contents.

2. **Organize the Data**:

   The dataset contains several files, including `movie_lines.txt` and `movie_conversations.txt`. You can use Python to process and organize the data. Here's an example of how to do this:

SOURCE CODE:

```python
# Import necessary libraries
import re
import os

# Define a function to load conversations
def load_conversations(filepath):
    conversations = []
    with open(filepath, 'r', encoding='iso-8859-1') as file:
        lines = file.readlines()
    for line in lines:
        conversations.append(line.strip().split(' +++$+++ ')[-1][1:-1])
    return conversations

# Define a function to load lines
def load_lines(filepath):
```

```
    lines = {}
    with open(filepath, 'r', encoding='iso-8859-1') as file:
        for line in file:
            parts = line.strip().split(' +++$+++ ')
            if len(parts) == 5:
                lines[parts[0]] = parts[4]
    return lines

# Define the paths to the dataset files
corpus_path = 'path/to/cornell_movie_dialogs_corpus'
lines_filepath = os.path.join(corpus_path, 'movie_lines.txt')
conversations_filepath = os.path.join(corpus_path, 'movie_conversations.txt')

# Load the lines and conversations
lines = load_lines(lines_filepath)
conversations = load_conversations(conversations_filepath)

# Process the data as needed
# For instance, you can create pairs of questions and answers
qa_pairs = []
for conversation in conversations:
    for i in range(len(conversation) - 1):
        input_line = lines[conversation[i]]
        target_line = lines[conversation[i + 1]]
        if input_line and target_line:
            qa_pairs.append((input_line, target_line))
```

```
# Print a sample QA pair

print(qa_pairs[0])
```

3. **Data Preprocessing**:

   Depending on your chatbot model, you may need to preprocess the data, including tokenization and padding.

4. **Training a Chatbot Model**:

   With the dataset organized and preprocessed, you can use it to train a chatbot model using deep learning libraries such as TensorFlow or PyTorch. There are various chatbot architectures to choose from, including seq2seq models and transformer-based models like GPT-3.

Creating a complete chatbot using Python with all the features and capabilities of a sophisticated chatbot is beyond the scope of a simple source code snippet. However, I can provide you with a basic example of a Python chatbot using the Rasa NLU and Rasa Core libraries, which is a popular open-source framework for building chatbots.

Please note that you need to install the Rasa NLU and Rasa Core libraries and set up the required configuration and training data to use this example.

1. **Install Rasa**:

   You can install Rasa by using pip:

   ```
   pip install rasa
   ```

2. **Create a Chatbot Project**:

You can use the Rasa command-line tool to create a new chatbot project:

```
rasa init
```

This will set up the directory structure for your chatbot project.

3. **Define Intents and Responses**:

In your project directory, you will find a `data/nlu.md` file where you can define user intents and example expressions. You can also specify responses in the `domain.yml` file.

4. **Train the NLU Model**:

Train the NLU (Natural Language Understanding) model using the provided training data:

```
rasa train nlu
```

5. **Train the Core Model**:

Train the core model to handle the conversation flow:

```
```

```
rasa train
```

6. **Run the Chatbot**:

   Start your chatbot using the following command:

   ```
   rasa shell
   ```

   Your chatbot will be running in the command line, and you can interact with it.

Here's an example of what the `data/nlu.md` and `domain.yml` files might look like:

nlu.md:

```markdown
## intent:greet
- hello
- hi
- hey

## intent:goodbye
- goodbye
- bye
- see you
```

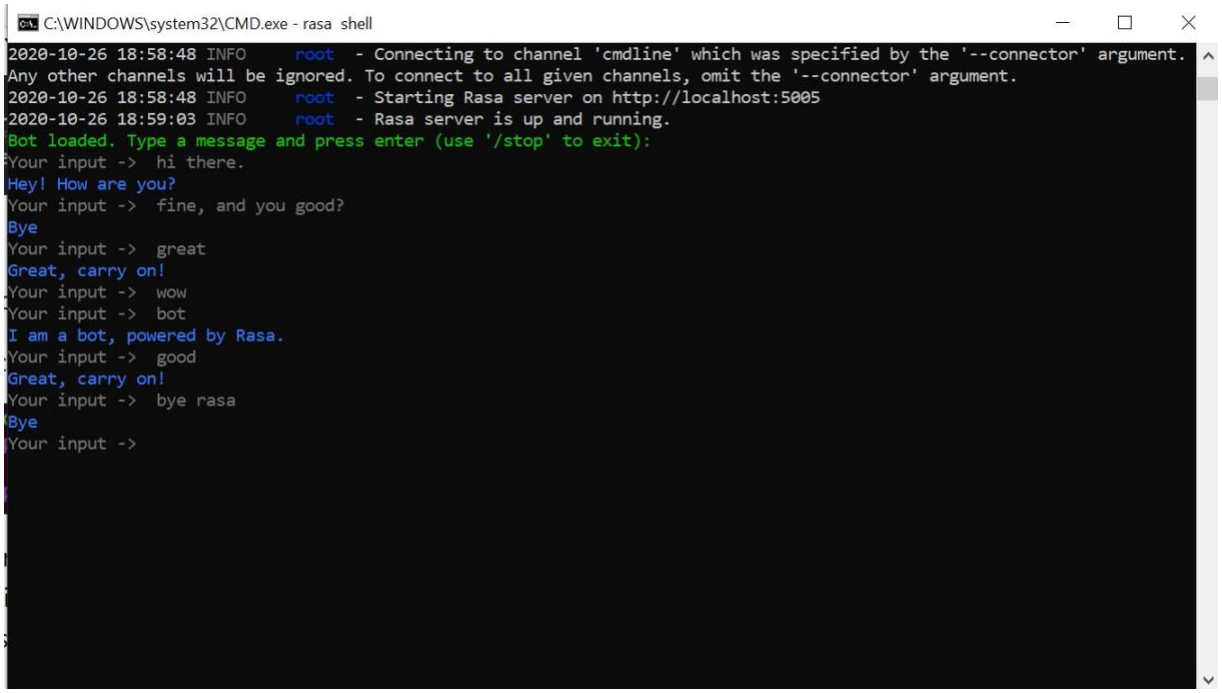
domain.yml:

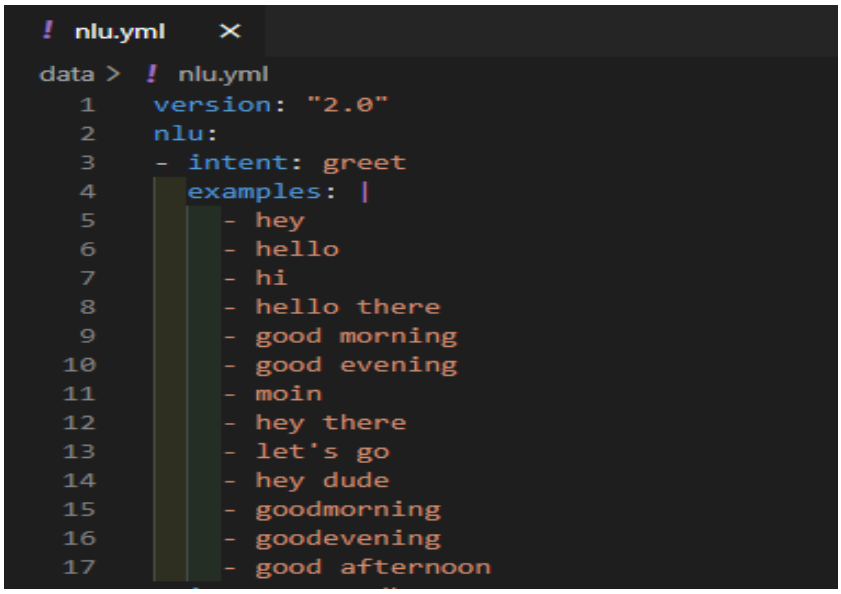
```yaml
intents:
  - greet
  - goodbye

responses:
  utter_greet:
    - text: "Hello! How can I assist you today?"

  utter_goodbye:
    - text: "Goodbye! Have a great day."
```

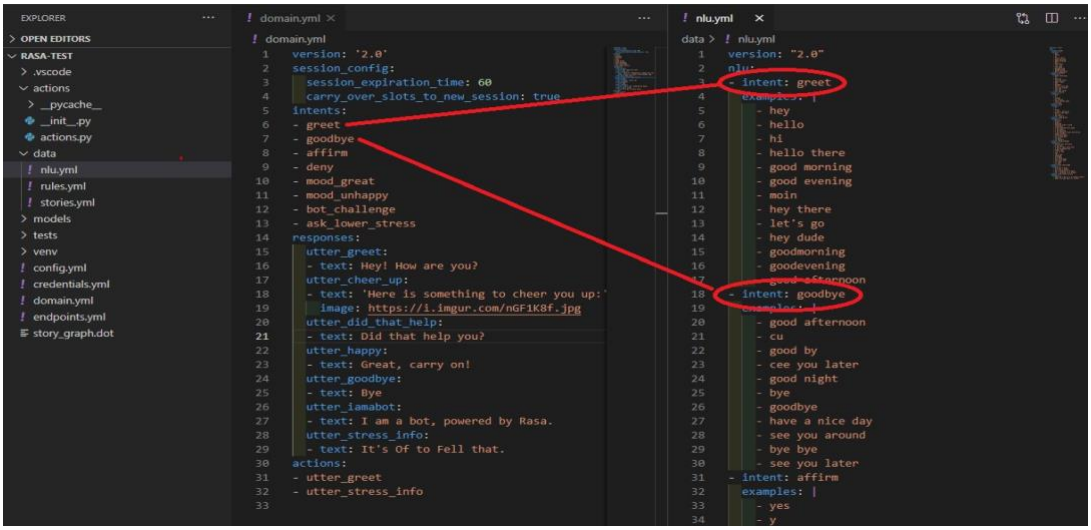
```
import random

# Define the chatbot's responses
responses = {
    "hello": "Hello! How can I assist you today?",
    "how are you": "I'm just a computer program, so I don't have feelings, but I'm here to help you!",
    "goodbye": "Goodbye! Have a great day.",
    "default": "I'm not sure how to respond to that. Please ask another question."
}

# Function to get a response from the chatbot
```

```
def get_response(user_input):
    user_input = user_input.lower()  # Convert user input to lowercase
    response = responses.get(user_input, responses["default"])
    return response

# Main chat loop
print("Chatbot: Hello! Type 'goodbye' to exit.")
while True:
    user_input = input("You: ")
    if user_input.lower() == 'goodbye':
        print("Chatbot: Goodbye!")
        break
    response = get_response(user_input)
    print("Chatbot:", response)
```

OUTPUT:







CONCLUSION:

This bot was built to respond to the inquiries of the Tawjihi students regarding each of the university's faculties and their specializations, with extracted information for each specialization, familiarizing students with the level exams that students submit about their enrollment in the university, introducing the educational

qualification diploma program and the mechanism for joining it. Giving students notes on the electronic enrollment application package, the locations of approved banks, and how to fill out the application. Introduce Bagrut students to the conditions and notes that must betaken into account in the event of joining Palestine Polytechnic University and the mechanism for calculating grades. Introduce students to the procedures followed to reserve a seat and what documents are required after the student is accepted. Introducing students to the system of transferring to Palestine Polytechnic University from another university on the undergraduate system. Informing students of the university's teaching system and language. Introducing students tothe student exchange system with other universities. Introducing students to the system of grants, exemptions, and financial aid provided to students. Informing students of cases in which the student loses his university seat. Introducing students to the installment refund system for new students and its conditions.