**AVS COLLEGE OF TECHNOLOGY**

**MENTOR NAME : MOHANA. P (623521106033)**

**PROJECT NAME: CREATE A CHATBOT IN PYTHON**

* **Introduction**:

“A chatbot in Python is a program that uses natural language processing and machine learning techniques to simulate human conversation. It interprets user input, processes it, and generates appropriate responses. Python provides various libraries such as NLTK, spaCy, and TensorFlow that can be used to develop chatbots capable of understanding and generating human-like text-based interactions.”

* **The core concepts of chatbot creation are –**

1. Natural Language Processing (NLP) – Chatbots use NLP to understand human language and interpret the user’s intent. NLP involves techniques like tokenization, part-of-speech tagging, and named entity recognition.
2. Dialog Management – Dialog management is responsible for managing the flow of the conversation and maintaining context across multiple turns of the conversation.
3. Machine Learning – Machine learning is used to train chatbots to recognize patterns in data, make predictions, and improve over time. Techniques like supervised learning, unsupervised learning, and reinforcement learning are used in chatbot development.
4. APIs and Integrations – Chatbots often need to integrate with external services and APIs to provide information or complete tasks for the user.
5. User Experience (UX) – The user experience is critical for chatbots, as they should be easy and intuitive to use. UX considerations include designing conversational flows, choosing appropriate response types, and providing clear and helpful feedback to the user.

* **Prerequisites**

Before we dive into the task few things should is expected to be installed onto your system –

**List of recommended settings –**

1. Pip install pandas, matplotlib
2. It is expected that the user will have access to any standalone IDE such as VS-Code, PyCharm, Atom or Sublime text.
3. Even online Python compilers can also be used such as Kaggle.com, Google Cloud platform or any other will do.
4. Updated version of Python. At the time of writing the article I have used 3.10.9 version.
5. Knowledge of the use of Jupyter notebook.
6. Knowledge and application of virtual environment would be beneficial but not required.
7. It is also expected that the person will have a good understanding of statistics and mathematics.
8. Installation of Python-NLTK(<http://www.nltk.org/install.html>).
9. Familiarity with Text processing (Tokenization, Lemma, Stemming).

* **Installing Required Libraries**

First, we need to install the required libraries for Developing a chatbot. NLTK, Regex, random and string libraries are required for chatbot development. To install these libraries, we can use pip command.

* !pip install nltk
* !pip install regex
* !pip install random
* !pip install string
* **Importing Required Libraries**

After installing the necessary libraries, we need to import these libraries in our python notebook. Below is the code for importing these libraries.

* + Import nltk
  + Import re
  + Import random
  + Import string
  + From string import punctuation
* **Preprocessing the Data**

Once the required packages are installed and imported, we need to preprocess the data. Preprocessing includes removing all the unnecessary data, tokenizing the data into sentences, and removing stopwords. Stopwords are the most common words that have little or no meaning in the context of the conversation, such as ‘a’, ‘is’ etc.

# Download stopwords from nltk

Nltk.download(‘punkt’)

Nltk.download(‘stopwords’)

Stop\_words = set(nltk.corpus.stopwords.words(‘english’))

Def sentence\_tokenizer(data):

# Function for Sentence Tokenization

Return nltk.sent\_tokenize(data.lower())

Def word\_tokenizer(data):

# Function for Word Tokenization

Return nltk.word\_tokenize(data.lower())

Def remove\_noise(word\_tokens):

# Function to remove stop words and punctuation

Cleaned\_tokens = []

For token in word\_tokens:

If token not in stop\_words and token not in punctuation:

Cleaned\_tokens.append(token)

* **Building a Chatbot**

Now that we have performed preprocessing on the data, we are ready to build the chatbot. The flow of the chatbot can be summarized in the following steps –

* Define the list of patterns and responses
* Initialize an infinite while loop
* Have the User Input a query
* Tokenize the query and remove stop words
* Match the query with one of the patterns and return a response.

# Define the Patterns and Responses

Patterns = [

(r’hi|hello|hey’, [‘Hi there!’, ‘Hello!’, ‘Hey!’]),

(r’bye|goodbye’, [‘Bye’, ‘Goodbye!’]),

(r’(\w+)’, [‘Yes, go on’, ‘Tell me more’, ‘I’m listening…’]),

(r’(\?)’, [‘I’m sorry, but I can’t answer that’,’Please ask me another question’, ‘I’m not sure what you mean.’])

]

# Function to generate response for the user input

Def generate\_response(user\_input):

# Append User Input to chat history

Conversation\_history.append(user\_input)

# Generate Random response

Response = random.choice(responses)

Return response

# Main loop of chatbot

Conversation\_history = []

Responses = [response for pattern, response in patterns]

While True:

# User Input

User\_input = input(“You: “)

# End the Loop if the User Says Bye or Goodbye

If user\_input.lower() in [‘bye’, ‘goodbye’]:

Print(‘Chatbot: Goodbye!’)

Break

# Tokenize the User Input

User\_input\_tokenized = word\_tokenizer(user\_input)

# Remove Stop Words

User\_input\_nostops = remove\_noise(user\_input\_tokenized)

# Process Query and Generate Response

Chatbot\_response = generate\_response(user\_input\_nostops)

# Print Response

Print(‘Chatbot:’, chatbot\_response)

* **Final program**

Import nltk

Import re

Import random

Import string

From string import punctuation

# Download stopwords from nltk

Nltk.download(‘punkt’)

Nltk.download(‘stopwords’)

Stop\_words = set(nltk.corpus.stopwords.words(‘english’))

Def sentence\_tokenizer(data):

# Function for Sentence Tokenization

Return nltk.sent\_tokenize(data.lower())

Def word\_tokenizer(data):

# Function for Word Tokenization

Return nltk.word\_tokenize(data.lower())

Def remove\_noise(word\_tokens):

# Function to remove stop words and punctuation

Cleaned\_tokens = []

For token in word\_tokens:

If token not in stop\_words and token not in punctuation:

Cleaned\_tokens.append(token)

Return cleaned\_tokens

# Define the Patterns and Responses

Patterns = [

(r’hi|hello|hey’, [‘Hi there!’, ‘Hello!’, ‘Hey!’]),

(r’bye|goodbye’, [‘Bye’, ‘Goodbye!’]),

(r’(\w+)’, [‘Yes, go on’, ‘Tell me more’, ‘I’m listening…’]),

(r’(\?)’, [‘I’m sorry, but I can’t answer that’, ‘Please ask me another question’, ‘I’m not sure what you mean.’])

]

# Function to generate response for the user input

Def generate\_response(user\_input):

# Append User Input to chat history

Conversation\_history.append(user\_input)

# Generate Random response

Response = random.choice(responses)

Return response

# Main loop of chatbot

Conversation\_history = []

Responses = [response for pattern, response in patterns]

While True:

# User Input

User\_input = input(“You: “)

# End the Loop if the User Says Bye or Goodbye

If user\_input.lower() in [‘bye’, ‘goodbye’]:

Print(‘Chatbot: Goodbye!’)

Break

# Tokenize the User Input

User\_input\_tokenized = word\_tokenizer(user\_input)

# Remove Stop Words

User\_input\_nostops = remove\_noise(user\_input\_tokenized)

# Process Query and Generate Response

Chatbot\_response = generate\_response(user\_input\_nostops)

# Print Response

Print(‘Chatbot:’, chatbot\_response)

* **Output**

In this section we can see the output of the code: User input −

Hi There

You: (Press ‘Enter’ to confirm or ‘Escape’ to cancel)

The user needs enter a string which is like a welcome message or a greeting, the chatbot will respond accordingly.

Chatbot: [‘Yes, go on’, ‘Tell me more’, ‘I’m listening…’]

Chatbot: [‘Bye’, ‘Goodbye!’]

Based on the response the chatbot will create response

Tell me more

You: (Press “Enter” to confirm or ‘Escape to cancel)

Chatbot: [‘Yes, go on’, ‘Tell me more’, ‘I’m listening…]

Chatbot: [‘Bye’, ‘Goodbye!’]

Bye

You: (Press ‘Enter to confirm or ‘Escape’ to cancel)

Chatbot: [‘Yes, go on’, ‘Tell me more’, ‘I’m listening…]

Chatbot: [‘Bye’, ‘Goodbye!’]

Chatbot: [‘Yes, go on’, ‘Tell me more’, ‘I’m listening…’]

Chatbot: Goodbye!

* **Conclusion :**

Creating a chatbot in Python can be a rewarding experience, allowing you to harness the power of natural language processing and machine learning to build a conversational interface. By leveraging libraries such as NLTK, spaCy, or Transformers, you can enable your chatbot to understand and generate human-like responses.

Implementing a robust and intuitive user interface and integrating it with a backend server can enhance the chatbot’s functionality. Regular updates and improvements are crucial to keeping the chatbot relevant and effective, ensuring a seamless user experience. By continually refining the chatbot’s capabilities and adapting to user feedback, you can create a sophisticated, intelligent, and user-friendly chatbot.