End to End Chatbot using Python

AMAN KHARWAL / MARCH 27, 2023 / MACHINE LEARNING / 5

A chatbot is a computer program that understands the intent of your query to answer with a solution. Chatbots are the most popular applications of Natural Language Processing in the industry. So, if you want to build an end-to-end chatbot, this article is for you. In this article, I will take you through how to create an end-to-end chatbot using Python.

What is an End to End Chatbot?

An end-to-end chatbot refers to a chatbot that can handle a complete conversation from start to finish without requiring human assistance. To create an end-to-end chatbot, you need to write a computer program that can understand user requests, generate appropriate responses, and take action when necessary. This involves collecting data, choosing a programming language and NLP tools, training the chatbot, and testing and refining it before making it available to users.

Once deployed, users can interact with the chatbot by sending it multiple requests and the chatbot can handle the entire conversation itself. To create an end-to-end chatbot using Python, we can follow the steps mentioned below:

1. Define Intents
2. Create training data
3. Train the chatbot
4. Build the chatbot
5. Test the chatbot
6. Deploy the chatbot

I hope you now have understood what an end-to-end chatbot is and the process of creating an end-to-end chatbot. In the section below, I'll walk you through how to build an end-to-end chatbot using Python.

End to End Chatbot using Python

Now let's start with creating an end-to-end chatbot using Python. I'll start this task by importing the necessary Python libraries for this task:

```
import os
import nltk
import ssl
import streamlit as st
```

```
import random
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

ssl._create_default_https_context = ssl._create_unverified_conte
nltk.data.path.append(os.path.abspath("nltk_data"))
nltk.download('punkt')
```

Now let's define some intents of the chatbot. You can add more intents to make the chatbot more helpful and more functional:

```
intents = [
    {
        "tag": "greeting",
        "patterns": ["Hi", "Hello", "Hey", "How are you", "What'
        "responses": ["Hi there", "Hello", "Hey", "I'm fine, tha
    },
    {
        "tag": "goodbye",
        "patterns": ["Bye", "See you later", "Goodbye", "Take ca
        "responses": ["Goodbye", "See you later", "Take care"]
    },
    {
        "tag": "thanks",
        "patterns": ["Thank you", "Thanks", "Thanks a lot", "I a
        "responses": ["You're welcome", "No problem", "Glad I co
    },
    {
        "tag": "about",
        "patterns": ["What can you do", "Who are you", "What are
        "responses": ["I am a chatbot", "My purpose is to assist
    },
    {
        "tag": "help",
        "patterns": ["Help", "I need help", "Can you help me",
```

```
        "responses": ["Sure, what do you need help with?", "I'm
    },
    {
        "tag": "age",
        "patterns": ["How old are you", "What's your age"],
        "responses": ["I don't have an age. I'm a chatbot.", "I
    },
    {
        "tag": "weather",
        "patterns": ["What's the weather like", "How's the weath
        "responses": ["I'm sorry, I cannot provide real-time wea
    },
    {
        "tag": "budget",
        "patterns": ["How can I make a budget", "What's a good b
        "responses": ["To make a budget, start by tracking your
    },
    {
        "tag": "credit_score",
        "patterns": ["What is a credit score", "How do I check m
        "responses": ["A credit score is a number that represent
    }
]
```

Now let's prepare the intents and train a Machine Learning model for the chatbot:

```
# Create the vectorizer and classifier
vectorizer = TfidfVectorizer()
clf = LogisticRegression(random_state=0, max_iter=10000)

# Preprocess the data
tags = []
patterns = []
for intent in intents:
    for pattern in intent['patterns']:
```

```
        tags.append(intent['tag'])
        patterns.append(pattern)

# training the model
x = vectorizer.fit_transform(patterns)
y = tags
clf.fit(x, y)
```

Now let's write a Python function to chat with the chatbot:

```
def chatbot(input_text):
    input_text = vectorizer.transform([input_text])
    tag = clf.predict(input_text)[0]
    for intent in intents:
        if intent['tag'] == tag:
            response = random.choice(intent['responses'])
            return response
```

Till now, we have created the chatbot. After running the code, you can interact with the chatbot in the terminal itself. To turn this chatbot into an end-to-end chatbot, we need to deploy it to interact with the chatbot using a user interface. To deploy the chatbot, I will use the **streamlit** library in Python, which provides amazing features to create a user interface for a Machine Learning application in just a few lines of code.

So, here's how we can deploy the chatbot using Python:

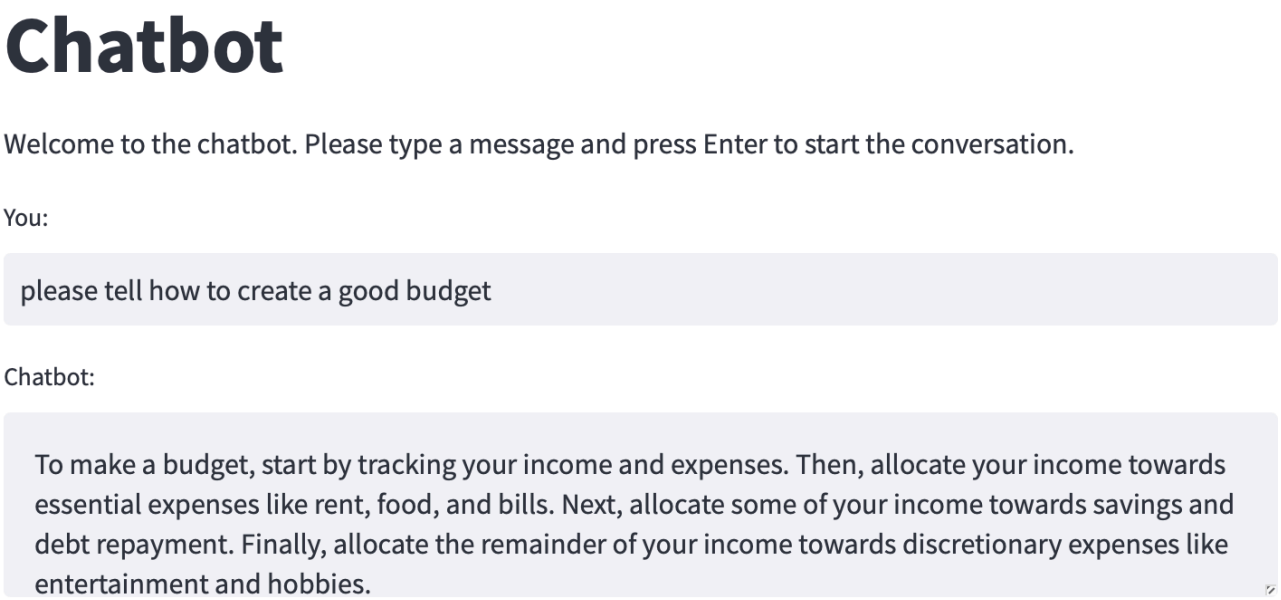
```
counter = 0

def main():
    global counter
    st.title("Chatbot")
    st.write("Welcome to the chatbot. Please type a message an
```

```
7
8       counter += 1
9       user_input = st.text_input("You:", key=f"user_input_{counter
10
11      if user_input:
12          response = chatbot(user_input)
13          st.text_area("Chatbot:", value=response, height=100, max
14
15          if response.lower() in ['goodbye', 'bye']:
16              st.write("Thank you for chatting with me. Have a gre
17              st.stop()
18
19  if __name__ == '__main__':
20      main()
```

To run this chatbot, use the command mentioned below in your terminal:

- streamlit run filename.py

After executing the command, you will see a UI of your chatbot in your web browser as shown below:



Output Video

So, this is how you can create an end-to-end chatbot using the Python programming language.

Summary

An end-to-end chatbot refers to a chatbot that can handle a complete conversation from start to finish without requiring human assistance. Creating an end-to-end Chatbot involves collecting data, choosing a programming language and NLP tools, training the chatbot, and testing and refining it before making it available to users. I hope you liked this article on building an end-to-end chatbot using Python. Feel free to ask valuable questions in the comments section below.

Aman Kharwal

Data Strategist at Statso. My aim is to decode data science for the real world in the most simple words.

Recommended For You

Text Classification Pipeline with Hugging Face Transformers

July 22, 2025

Deep Learning & NLP Project Ideas

June 18, 2025

50+ Machine Learning Projects with Python

May 17, 2025

NLP & LLM Project Ideas for Resume

May 14, 2025

5 Comments

Gavin W Lampkin MARCH 27, 2023 / 9:57 PM REPLY

Very interesting project! I was wondering if you would always have to input more data into the database manually for the bot to have more information to regurgitate? Is there a section I miss understood where you implemented such function? Thanks so much for sharing, I love browsing your projects and website!

Aman Kharwal MARCH 27, 2023 / 10:25 PM REPLY

Yes, this is how Machine Learning works, the more data you have, the better your model will perform. It's like the brain of a human, the more you consume knowledge the more you can make decision based on your knowledge.

Joe Mama FEBRUARY 11, 2024 / 3:19 AM REPLY

This helped me a lot thank you for this.

Rohith APRIL 13, 2025 / 6:57 AM REPLY

Please build a chatbot using Transformer

Fama Cisse JANUARY 21, 2025 / 10:03 PM REPLY

Thank you so much