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:

```
1 import os
2 import nltk
3 import ssl
4 import streamlit as st
5 import random
6 from sklearn.feature_extraction.text import TfidfVe
```

```
from sklearn.linear_model import LogisticRegressior

ssl._create_default_https_context = ssl._create_unv
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
        "responses": ["Hi there", "Hello", "Hey", '
    },
    {
        "tag": "goodbye",
        "patterns": ["Bye", "See you later", "Goodb
        "responses": ["Goodbye", "See you later", '
    },
    {
        "tag": "thanks",
        "patterns": ["Thank you", "Thanks", "Thanks
        "responses": ["You're welcome", "No problem
    },
    {
        "tag": "about",
        "patterns": ["What can you do", "Who are yo
        "responses": ["I am a chatbot", "My purpose
    },
    {
        "tag": "help",
```

```
        "patterns": ["Help", "I need help", "Can yo
        "responses": ["Sure, what do you need help
    },
    {
        "tag": "age",
        "patterns": ["How old are you", "What's you
        "responses": ["I don't have an age. I'm a c
    },
    {
        "tag": "weather",
        "patterns": ["What's the weather like", "Ho
        "responses": ["I'm sorry, I cannot provide
    },
    {
        "tag": "budget",
        "patterns": ["How can I make a budget", "Wh
        "responses": ["To make a budget, start by t
    },
    {
        "tag": "credit_score",
        "patterns": ["What is a credit score", "How
        "responses": ["A credit score is a number t
    }
]
```

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=1
```

```
# 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['respon
            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.

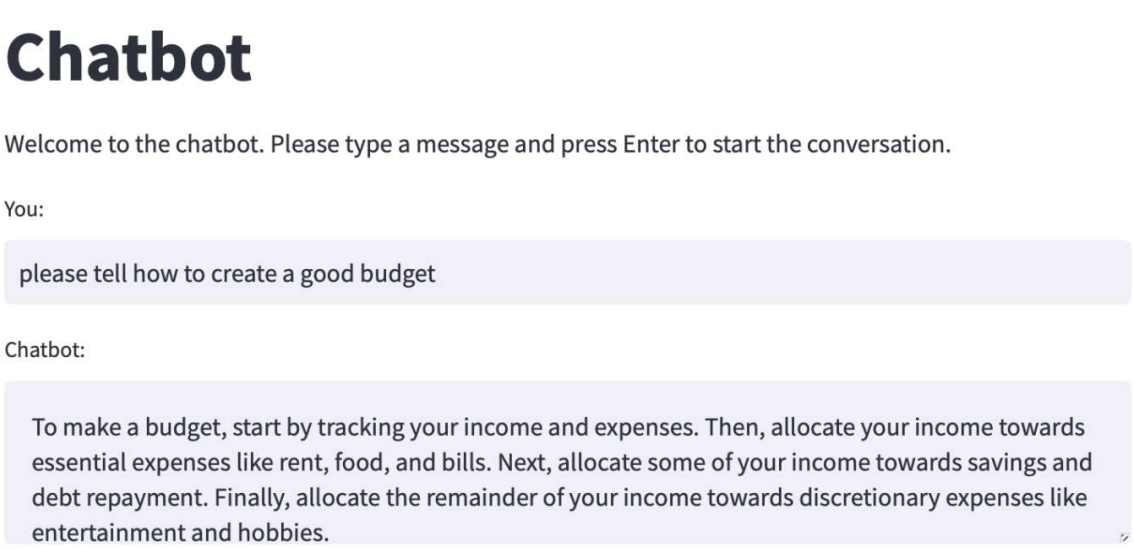
So, here's how we can deploy the chatbot using Python:

```
unter = 0

f main():
 global counter
 st.title("Chatbot")
 st.write("Welcome to the chatbot. Please type a mess

 counter += 1
 user_input = st.text_input("You:", key=f"user_input_

if user_input:
    response = chatbot(user_input)
    st.text_area("Chatbot:", value=response, height=

    if response.lower() in ['goodbye', 'bye']:
        st.write("Thank you for chatting with me. Ha
        st.stop()

__name__ == '__main__':
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.