SURYA GROUP OF INSTITUTIONS

NAAN MUDHALVAN

IBM-ARTIFICIAL INTELLIGENCE

GNANAVEL B

422221104012

Creating a Chatbot In Python

TEAM : 03

Creating a Chatbot with Machine Learning in Python

Introduction

Chatbots are becoming increasingly popular and are being used in various applications, from customer service to personal assistants. With the advancement in technology, creating a chatbot has become easier than ever before. In this article, we will learn how to create a chatbot in Python, step-by-step.

Before we dive into coding, let's discuss what a chatbot is and how it works. A chatbot is a program designed to mimic human conversations. It uses natural language processing (NLP) and machine learning algorithms to understand user input and respond accordingly.

To create a chatbot, we will be using Python and the following libraries:

- NLTK (Natural Language Toolkit): a library for NLP tasks such as tokenization, stemming, and parsing.
- TensorFlow: a library for building and training machine

learning models.

Keras: a high-level API built on top of TensorFlow for building neural networks.

Step 1: Importing Libraries

First, let's import the necessary libraries:

```
import nltk

from nltk.stem import WordNetLemmatizer

import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers
```

Step 2: Preparing Data

Next, we will prepare the data for our chatbot. We will create a list of intents, where each intent represents a possible conversation with the bot. For example, we can have an intent for greeting the bot, an intent for asking for help, and an intent for saying goodbye.

```
intents = [

  {

      "tag": "greeting",

      "patterns": ["Hi", "Hello", "Hey", "How are you", "What's up"],

      "responses": ["Hi there", "Hello", "I'm good, how are you?", "Not much"]
```

```
    },
    {
        "tag": "help",
        "patterns": ["Can you help me", "What can you do", "How do I use this"],
        "responses": ["Sure, what do you need help with?", "I can do a lot of things, just ask me", "Here are some commands you can use"]
    },
    {
        "tag": "goodbye",
        "patterns": ["Bye", "See you later", "Goodbye"],
        "responses": ["Goodbye", "See you later", "Have a nice day"]
    }
]
```

We will also create a list of all the words used in the intents, called the vocabulary. This will be used to represent each sentence as a vector of numbers.

```
words = []
for intent in intents:
    for pattern in intent["patterns"]:
        words.extend(nltk.word_tokenize(pattern))
words = sorted(set(words))
```

Step 3: Preprocessing Data

Before we can feed the data into our neural network, we need to preprocess it. This involves tokenizing the sentences, lemmatizing the words (converting them to their base form), and converting the sentences to vectors.

```
lemmatizer = WordNetLemmatizer()


def preprocess(sentence):

   tokens = nltk.word_tokenize(sentence)

   lemmatized_tokens = [lemmatizer.lemmatize(token.lower()) for token in
tokens]

   vector = np.zeros(len(words))

   for word in lemmatized_tokens:

      if word in words:

         vector[words.index(word)] = 1

   return vector
```

Step 4: Creating Model

Now, we will create a neural network model to classify the user input into one of the intents. We will use a simple model with one hidden layer.

```
model = keras.Sequential(

   [

      layers.Dense(8, input_shape

layers.Dense(len(intents), activation="softmax")

   ]

)
```

```
model.compile(loss="categorical_crossentropy", optimizer="adam")
```

.

Step 5: Training Model:

Now, we will train the model on our dataset. We will use the preprocessing function we created earlier to convert each sentence into a vector, and we will one-hot encode the intents.

```
X_train = np.array([preprocess(pattern) for intent in intents for pattern in intent["patterns"]])

y_train = keras.utils.to_categorical([i for i, intent in enumerate(intents) for pattern in intent["patterns"]], num_classes=len(intents))


model.fit(X_train, y_train, epochs=1000, verbose=1)
```

.

Step 6: Testing Model:

Once the model is trained, we can use it to classify new user input. We will define a function that takes a user input, preprocesses it, and predicts the intent using the trained model.

```
def predict_intent(sentence):

   X = np.array([preprocess(sentence)])

   y_pred = model.predict(X)

   return intents[np.argmax(y_pred)]["tag"]
```

.

Step 7: Interacting with Chatbot

Finally, we will create a loop to interact with the chatbot. We will take user input and use the predict_intent function to classify it. We will then randomly select a response from the intent and print it to the user.

```
while True:

    user_input = input("You: ")

    intent = predict_intent(user_input)

    response = np.random.choice(intents[intent]["responses"])

    print("Bot: " + response)
```

Here is the complete code:

```
import nltk

from nltk.stem import WordNetLemmatizer

import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers


intents = [

    {

        "tag": "greeting",

        "patterns": ["Hi", "Hello", "Hey", "How are you", "What's up"],

        "responses": ["Hi there", "Hello", "I'm good, how are you?", "Not
much"]
```

```
  },
  {
    "tag": "help",
    "patterns": ["Can you help me", "What can you do", "How do I use
this"],
    "responses": ["Sure, what do you need help with?", "I can do a lot of
things, just ask me", "Here are some commands you can use"]
  },
  {
    "tag": "goodbye",
    "patterns": ["Bye", "See you later", "Goodbye"],
    "responses": ["Goodbye", "See you later", "Have a nice day"]
  }
]

words = []
for intent in intents:
  for pattern in intent["patterns"]:
    words.extend(nltk.word_tokenize(pattern))
words = sorted(set(words))

lemmatizer = WordNetLemmatizer()

def preprocess(sentence):
    tokens = nltk.word_tokenize(sentence)
    lemmatized_tokens = [lemmatizer.lemmatize(token.lower()) for token in
tokens]
    vector = np.zeros(len(words))
    for word in lemmatized_tokens:
        if word in words:
            vector[words.index(word)] = 1
    return vector


model = keras.Sequential(
    [


layers.InputLayer(input_shape=(len(words),)),
    layers.Dense(8, activation="relu"),
    layers.Dense(8, activation="relu"),
    layers.Dense(len(intents), activation="softmax")
]


 )


model.compile(loss="categorical_crossentropy", optimizer="adam")


X_train = np.array([preprocess(pattern) for intent in intents for pattern in
intent["patterns"]])

y_train = keras.utils.to_categorical([i for i, intent in enumerate(intents) for
pattern in intent["patterns"]], num_classes=len(intents))


model.fit(X_train, y_train, epochs=1000, verbose=1)



def predict_intent(sentence):

X = np.array([preprocess(sentence)])

y_pred = model.predict(X)

return intents[np.argmax(y_pred)]["tag"]


while True:

user_input = input("You: ")

intent = predict_intent(user_input)

response = np.random.choice(intents[intent]["responses"])

print("Bot: " + response)
```

Conclusion:

In this article, we have learned how to create a simple chatbot using Python and natural language processing techniques.

We have used the NLTK library to preprocess the user input and the Keras library to build and train a simple neural network model. While this chatbot is far from perfect, it provides a solid foundation for building more complex chatbots that can understand natural language and respond appropriately.

With additional training data and more sophisticated techniques, you can build a chatbot that can perform a wide range of tasks and engage in natural conversations with users.