ChatBot using Machine Learning

CA-3(INT248)

Index

Submitted by:
Jagarlamudi Lakshmi Narayana
KMO73
B.Tech(CSE)
11808647

Submitted to:
Prof.Ankitha Wadhawan
Lovely Professional University
Phagwara, Punjab, India.

I. How Does a Chatbot Work?

Since we will be developing a Chatbot with Python using Machine Learning, we need some data to train our model. But we're not going to collect or download a large dataset since this is just a chatbot. We can just create our own dataset to train the model.

To create this dataset to create a chatbot with Python, we need to understand what intents we are going to train. An "intention" is the user's intention to interact with a chatbot or the intention behind every message the chatbot receives from a particular user.

Therefore, it is important to understand the good intentions of your chatbot depending on the domain you will be working with. So why does he need to define these intentions? This is a very important point to understand.

In order to answer questions asked by the users and perform various other tasks to continue conversations with the users, the chatbot really needs to understand what users are saying or having 'intention to do. This is why your chatbot must understand the intentions behind users' messages.

How can you get your chatbot to understand the intentions so that users feel like they know what they want and provide accurate answers? The strategy here is to set different intents and create training samples for those intents and train your chatbot model with these sample training data as model training data (X) and intents in as model training categories (Y).

II. Requirements

To create a chatbot with Python and Machine Learning, you need to install some packages. All the packages you need to install to create a chatbot with Machine Learning using the Python programming language are mentioned below:

- Tensorflow
- nltk
- colorama
- numpy
- scikit learn
- Flask

III. Defining the intents of a chatbot

Now we need to define a few simple intents and a group of messages that match those intents and also map some responses based on each intent category. I'll create a JSON file named "intents.json" including this data as follows:

```
{"intents": [
    {"tag": "greeting",
     "patterns": ["Hi", "Hey", "Is anyone there?", "Hello", "Hay"],
     "responses": ["Hello", "Hi", "Hi there"]
    },
    {"tag": "goodbye",
     "patterns": ["Bye", "See you later", "Goodbye"],
     "responses": ["See you later", "Have a nice day", "Bye! Come back
again"]
    },
    {"tag": "thanks",
     "patterns": ["Thanks", "Thank you", "That's helpful", "Thanks for the
help"],
     "responses": ["Happy to help!", "Any time!", "My pleasure", "You're
most welcome!"]
    },
    {"tag": "about",
     "patterns": ["Who are you?", "What are you?", "Who you are?" ],
     "responses": ["I.m Joana, your bot assistant", "I'm Joana, an
Artificial Intelligent bot"]
    },
    {"tag": "name",
    "patterns": ["what is your name", "what should I call you", "whats your
name?"],
    "responses": ["You can call me Joana.", "I'm Joana!", "Just call me as
Joana"]
    },
    {"tag": "help",
    "patterns": ["Could you help me?", "give me a hand please", "Can you
help?", "What can you do for me?", "I need a support", "I need a help",
"support me please"],
    "responses": ["Tell me how can assist you", "Tell me your problem to
assist you", "Yes Sure, How can I support you"]
    },
    {"tag": "createaccount",
    "patterns": ["I need to create a new account", "how to open a new
account", "I want to create an account", "can you create an account for me",
"how to open a new account"],
```

```
    "responses": ["You can just easily create a new account from our web
site", "Just go to our web site and follow the guidelines to create a new
account"]
    },
    {"tag": "complaint",
    "patterns": ["have a complaint", "I want to raise a complaint", "there
is a complaint about a service"],
    "responses": ["Please provide us your complaint in order to assist you",
"Please mention your complaint, we will reach you and sorry for any
inconvenience caused"]
    }
]
}
```

IV. Data preparation:

The second step of this task to create a chatbot with Python and Machine Learning is to prepare the data to train our chatbot. I'll start this step by importing the necessary libraries and packages:

```
import json
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, GlobalAveragePooling1D
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import LabelEncoder
```

Now I will read the JSON file and process the required files:

```
with open('intents.json') as file:
    data = json.load(file)

training_sentences = []
training_labels = []
labels = []
responses = []


for intent in data['intents']:
    for pattern in intent['patterns']:
        training_sentences.append(pattern)
        training_labels.append(intent['tag'])
    responses.append(intent['responses'])

    if intent['tag'] not in labels:
        labels.append(intent['tag'])

num_classes = len(labels)
```

Now we need to use the label encoder method provided by the Scikit-Learn library in Python:

```
lbl_encoder = LabelEncoder()
lbl_encoder.fit(training_labels)
training_labels = lbl_encoder.transform(training_labels)
```

V. Tokenization:

```
vocab_size = 1000
embedding_dim = 16
max_len = 20
oov_token = "<OOV>"

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_token)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(training_sentences)
padded_sequences = pad_sequences(sequences, truncating='post',
maxlen=max_len)
```

VI. Training a Neural network:

Now the next and most important step in the process of building a chatbot with Python and Machine Learning is to train a neural network. Now, I will train and create a neural network to train our chatbot:

```
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, input_length=max_len))
model.add(GlobalAveragePooling1D())
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

model.summary()
epochs = 500
history = model.fit(padded_sequences, np.array(training_labels),
epochs=epochs)
```

VII. Saving the Neural network:

```
# to save the trained model
model.save("chat_model")

import pickle

# to save the fitted tokenizer
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

```
# to save the fitted label encoder
with open('label_encoder.pickle', 'wb') as ecn_file:
    pickle.dump(lbl_encoder, ecn_file, protocol=pickle.HIGHEST_PROTOCOL)
```

VIII. Building The ChatBot:

Now I am going to implement a chat function to interact with a real user. When the message from the user will be received, the chatbot will compute the similarity between the sequence of the new text and the training data.

Taking into account the trust scores obtained for each category, it categorizes the user's message according to an intention with the highest trust score:

```
import json
import numpy as np
from tensorflow import keras
from sklearn.preprocessing import LabelEncoder

import colorama
colorama.init()
from colorama import Fore, Style, Back

import random
import pickle

with open("intents.json") as file:
    data = json.load(file)


def chat():
    # load trained model
    model = keras.models.load_model('chat_model')

    # load tokenizer object
    with open('tokenizer.pickle', 'rb') as handle:
        tokenizer = pickle.load(handle)

    # load label encoder object
    with open('label_encoder.pickle', 'rb') as enc:
        lbl_encoder = pickle.load(enc)

    # parameters
    max_len = 20

    while True:
        print(Fore.LIGHTBLUE_EX + "User: " + Style.RESET_ALL, end="")
        inp = input()
        if inp.lower() == "quit":
            break

        result =
model.predict(keras.preprocessing.sequence.pad_sequences(tokenizer.texts_to_
sequences([inp]),
                                             truncating='post',
maxlen=max_len))
        tag = lbl_encoder.inverse_transform([np.argmax(result)])
```

```
        for i in data['intents']:
            if i['tag'] == tag:
                print(Fore.GREEN + "ChatBot:" + Style.RESET_ALL ,
np.random.choice(i['responses']))

        # print(Fore.GREEN + "ChatBot:" +
Style.RESET_ALL,random.choice(responses))

print(Fore.YELLOW + "Start messaging with the bot (type quit to stop)!" +
Style.RESET_ALL)
chat()
```