# CHAT BOT IN PYTHON

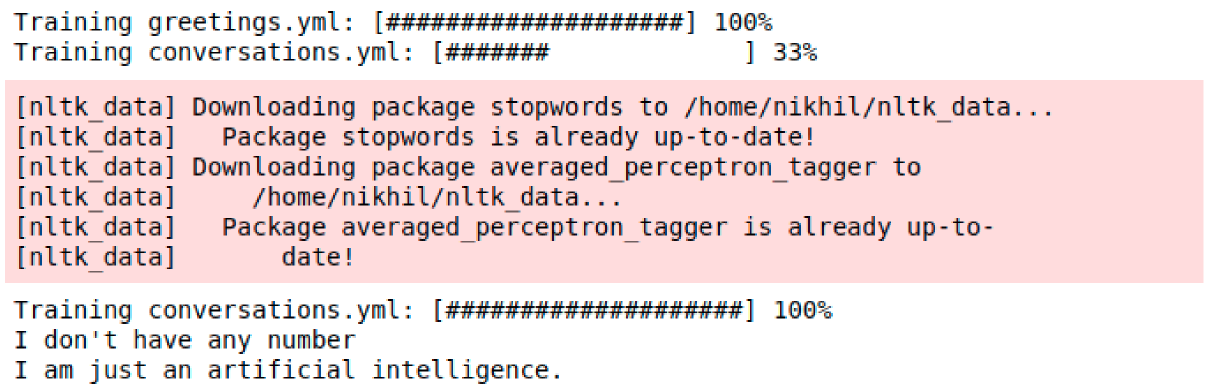
## Installation:

```
pip install chatter bot
```

```
 import numpy as np

import time

import os

from transformers import AutoModelForCausalLM,
AutoTokenizer

import torch



# Import "chatbot" from

# chatterbot package.

from chatterbot import ChatBot



# Inorder to train our bot, we have

# to import a trainer package

# "ChatterBotCorpusTrainer"

from chatterbot.trainers import ChatterBotCorpusTrainer
```

```
# Give a name to the chatbot “corona bot”

# and assign a trainer component.

chatbot=ChatBot('corona bot')


# Create a new trainer for the chatbot

trainer = ChatterBotCorpusTrainer(chatbot)


# Now let us train our bot with multiple corpus

trainer.train("chatterbot.corpus.english.greetings",

              "chatterbot.corpus.english.conversations" )


response = chatbot.get_response('What is your Number')

print(response)
```

## output:



```
Training greetings.yml: [####################] 100%
Training conversations.yml: [#######             ] 33%
[nltk_data] Downloading package stopwords to /home/nikhil/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/nikhil/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
Training conversations.yml: [####################] 100%
I don't have any number
I am just an artificial intelligence.
```

```
# Build a ChatBot class with all necessary modules to
make a complete conversation
class ChatBot():
    # initialize
    def __init__(self):
        # once chat starts, the history will be stored for
chat continuity
        self.chat_history_ids = None
        # make input ids global to use them anywhere
within the object
        self.bot_input_ids = None
        # a flag to check whether to end the conversation
        self.end_chat = False
        # greet while starting
        self.welcome()

    def welcome(self):
        print("Initializing ChatBot ...")
        # some time to get user ready
        time.sleep(2)
        print('Type "bye" or "quit" or "exit" to end chat
\n')
        # give time to read what has been printed
        time.sleep(3)
        # Greet and introduce
        greeting = np.random.choice([
            "Welcome, I am ChatBot, here for your kind
service",
            "Hey, Great day! I am your virtual assistant",
            "Hello, it's my pleasure meeting you",
            "Hi, I am a ChatBot. Let's chat!"
        ])
        print("ChatBot >>  " + greeting)

    def user_input(self):
        # receive input from user
        text = input("User    >> ")
        # end conversation if user wishes so
```

```
      if text.lower().strip() in ['bye', 'quit', 'exit']:
          # turn flag on
          self.end_chat=True
          # a closing comment
          print('ChatBot >>  See you soon! Bye!')
          time.sleep(1)
          print('\nQuitting ChatBot ...')
      else:
          # continue chat, preprocess input text
          # encode the new user input, add the
  else:
          # continue chat, preprocess input text
          # encode the new user input, add the
  eos_token and return a tensor in Pytorch
          self.new_user_input_ids =
  tokenizer.encode(text + tokenizer.eos_token, \
                                            return_tensors='pt')


    def bot_response(self):
      # append the new user input tokens to the chat
  history
      # if chat has already begun
      if self.chat_history_ids is not None:
          self.bot_input_ids =
  torch.cat([self.chat_history_ids,
  self.new_user_input_ids], dim=-1)
      else:
          # if first entry, initialize bot_input_ids
          self.bot_input_ids = self.new_user_input_ids


      # define the new chat_history_ids based on the
  preceding chats
      # generated a response while limiting the total
  chat history to 1000 tokens,
      self.chat_history_ids =
  model.generate(self.bot_input_ids,
  max_length=1000, \

  pad_token_id=tokenizer.eos_token_id)
```

```
        # last ouput tokens from bot
        response = 
tokenizer.decode(self.chat_history_ids[:, 
self.bot_input_ids.shape[-1]:][0], \
                        skip_special_tokens=True)
        # in case, bot fails to answer
        if response == "":
            response = self.random_response()
        # print bot response
        print('ChatBot >>  '+ response)

    # in case there is no response from model
    def random_response(self):
        i = -1
        response = 
tokenizer.decode(self.chat_history_ids[:, 
self.bot_input_ids.shape[i]:][0], \
                        skip_special_tokens=True)
        # iterate over history backwards to find the last
    token
        while response == '':
            i = i-1
            response =
    tokenizer.decode(self.chat_history_ids[:,
    self.bot_input_ids.shape[i]:][0], \
                           skip_special_tokens=True)
        # if it is a question, answer suitably
        if response.strip() == '?':
            reply = np.random.choice(["I don't know",
                                "I am not sure"])
        # not a question? answer suitably
        else:
            reply = np.random.choice(["Great",
                                 "Fine. What's up?",
                                 "Okay"
                                ])
        return reply
```

```
Initializing ChatBot ...
Type "bye" or "quit" or "exit" to end chat


ChatBot >>  Welcome, I am ChatBot, here for your
kind service
ChatBot >>  I'm good, how are you?
ChatBot >>  I do.
ChatBot >>  I don't really cook.
ChatBot >>  I don't really eat food.
ChatBot >>  I like that.
ChatBot >>  I like coffee.
ChatBot >>  I don't drink coffee.
ChatBot >>  I don't drink coffee
ChatBot >>  Fine. What's up?
ChatBot >>  Okay
ChatBot >>  See you soon! Bye!


Quitting  chat bot
```