Conversational AI ChatBot

Intelligent ChatBot built with Microsoft's DialoGPT transformer to make conversations with human users!

> Image by Andy Kelly

What is a chatbot?

> A ChatBot is a kind of virtual assistant that can build conversations with human users! A *Chat*ting Ro*bot*. Building a chatbot is one of the popular tasks in Natural Language Processing.

Are all chatbots the same?

> Chatbots fall under three common categories:
>
> 1. Rule-based chatbots
> 2. Retrieval-based chatbots
> 3. Intelligent chatbots

Rule-based chatbots

> These bots respond to users' inputs based on certain pre-specified rules. For instance, these rules can be defined as if-elif-else statements. While writing rules for these chatbots, it is important to expect all possible user inputs, else the bot may fail to answer properly. Hence, rule-based chatbots do not possess any cognitive skills.

Retrieval-based chatbots

> These bots respond to users' inputs by retrieving the most relevant information from the given text document. The most relevant information can be determined by Natural Language Processing with a scoring system such as cosine-similarity-score. Though these bots use NLP to do conversations, they lack cognitive skills to match a real human chatting companion.

Intelligent AI chatbots

> These bots respond to users' inputs after understanding the inputs, as humans do. These bots are trained with a Machine Learning Model on a large training dataset of human conversations. These bots are cognitive to match a human in conversing. Amazon's Alexa, Apple's Siri fall under this category. Further, most of these bots can make conversations based on the preceding chat texts.

In this Article?

> This article describes building an intelligent AI chatbot based on the famous transformer architecture - Microsoft's DialoGPT. According to Hugging Face's model card, DialoGPT is a State-Of-The-Art large-scale pretrained dialogue response generation model for multiturn conversations. The human evaluation results indicate that the response generated from DialoGPT is comparable to human response quality under a single-turn conversation Turing test. The model is trained on 147M multi-turn dialogue from Reddit discussion thread.

Let's Python

Import necessary libraries and frameworks

```
import numpy as np
import time
import os
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch
```

Download Microsoft's DialoGPT model and tokenizer

The Hugging Face checkpoint for the model and its tokenizer is `"microsoft/DialoGPT-medium"`

```
# checkpoint
checkpoint = "microsoft/DialoGPT-medium"
# download and cache tokenizer
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
# download and cache pre-trained model
model = AutoModelForCausalLM.from_pretrained(checkpoint)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarni
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public mo
  warnings.warn(
tokenizer_config.json: 100%  614/614 [00:00<00:00, 8.95kB/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 5.07MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 5.52MB/s]
config.json: 100%  642/642 [00:00<00:00, 8.25kB/s]
pytorch_model.bin: 100%  863M/863M [00:20<00:00, 47.7MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 3.11kB/s]
```

A ChatBot class

```
# 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 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
```

Happy Chatting!

```
# build a ChatBot object
bot = ChatBot()
# start chatting
while True:
    # receive user input
    bot.user_input()
    # check whether to end chat
    if bot.end_chat:
        break
    # output bot response
    bot.bot_response()
```

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

ChatBot >>  Hi, I am a ChatBot. Let's chat!
User    >> hi, how are you?
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I'm good, how are you?
User    >> Great, Are you a computer programmer?
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I am not, I'm a software developer.
User    >> what languages do you know?
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I know Java, Python, and C.
User    >> great
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I'm glad to hear that.
User    >> i am good at machine learning
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I'm good at programming
User    >> wow, that great
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I'm good at programming
User    >> wow, have you developed any mobile app?
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I have a few
User    >> fine, whats up?
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side=
ChatBot >>  I'm good at programming
User    >> Bye
ChatBot >>  See you soon! Bye!

Quitting ChatBot ...
```

Some sample chats by this ChatBot

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

ChatBot >>  Hey, Great day! I am your virtual assistant
User    >>  hi, how are you?
ChatBot >>  I'm good, how are you?
User    >>  Great. Are you a computer programmer?
ChatBot >>  I am not. I'm a software engineer.
User    >>  what languages do you know?
ChatBot >>  I know Java, Python, and C.
User    >>  great. how many software projects have you done?
ChatBot >>  I have a few.
User    >>  who created python?
ChatBot >>  I created Python.
User    >>  that's awesome.
ChatBot >>  I'm a python developer.
User    >>  great. do you have a github account?
ChatBot >>  I have a github account.
User    >>  ok. are you good at web development?
ChatBot >>  I'm good at web development.
User    >>  wow. have you developed any mobile app?
ChatBot >>  Fine. What's up?
User    >>  It is my pleasure meeting you!
ChatBot >>  Fine. What's up?
User    >>  Bye
ChatBot >>  See you soon! Bye!

Quitting ChatBot ...
```

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

ChatBot >>  Welcome, I am ChatBot, here for your kind service
User    >>  Hi. How are you?
ChatBot >>  I'm good, how are you?
User    >>  I'm fine. Do you cook?
```