In [1]:
```
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"`

In [2]:
```
# 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)
```

```
Downloading:   0%|          | 0.00/26.0 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/642 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/0.99M [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/446k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/823M [00:00<?, ?B/s]
```

A ChatBot class

In [3]:
```
# 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 Pytorc
            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_
        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[-
                                    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
                                    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.sha
                                    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!

In [4]:
```
# 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!

ChatBot >>  Hey there!

ChatBot >>  I do!

ChatBot >>  I don't know.

ChatBot >>  I was lying.

ChatBot >>  I'm not lying.

ChatBot >>  It's now

ChatBot >>  It's still sunny

ChatBot >>  I'm in Houston

ChatBot >>  I'm in Houston

ChatBot >>  I'm in Houston

ChatBot >>  I'm in Houston

ChatBot >>  I

ChatBot >>  Fine. What's up?

ChatBot >>  Fine. What's up?

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?
ChatBot >>  I do.
User    >>  What is your favourite recipe?
ChatBot >>  I don't really cook.
User    >>  No problem. What is your favourite food?
ChatBot >>  I don't really eat food.
User    >>  I know. I like sea food a lot.
ChatBot >>  I like that.
User    >>  I have coffee often. How about you?
ChatBot >>  I like coffee.
User    >>  Do you take coffee with milk and sugar?
ChatBot >>  I don't drink coffee.
User    >>  Oh, you like it but don't drink. I understand.
ChatBot >>  I don't drink coffee
User    >>  Shall we have dinner tonight?
ChatBot >>  Fine. What's up?
User    >>  I will book a table and inform you. Be ready.
ChatBot >>  Okay
User    >>  bye
ChatBot >>  See you soon! Bye!

Quitting ChatBot ...
```

Thank you for your valuable time!

Find this notebook on Kaggle here: https://www.kaggle.com/rajkumarl/conversational-ai-chatbot