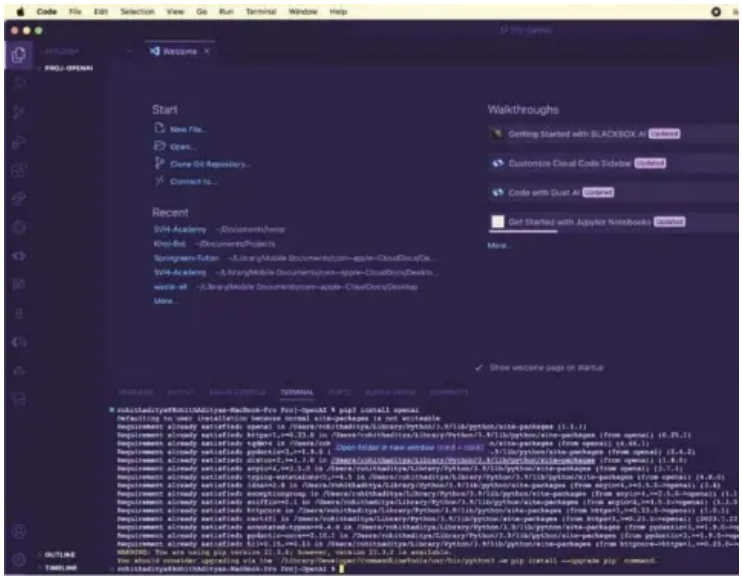
- Artificial Intelligence (AI)

A Deep Dive into GPT-3

Generative AI, particularly OpenAI's GPT-3, has revolutionised the landscape of artificial intelligence and natural language processing. Here we explore the capabilities of GPT-3 and the broader implications of open source generative AI.

Open Source for you · 1 Mar 2024 · 4 · By: S.V. Rohith

Generative AI is very exciting technology. It's like having a robot apprentice who can learn how to make art, music, and more! However, it still takes a lot of data, computing power, and learning for AI to match the creative abilities of people. But generative AI is getting better every day as it studies more examples of human creativity. The most amazing thing is that AI is not just copying work made by humans. It is able to take what it has learned and come up with new patterns, combinations, and innovations. Generative AI will help make art and content that has never been seen before!



Open source generative AI

The code used to create open source generative AI is available for everyone to look at, change, and use to build new things. It is not kept a secret by one company. Open source is about sharing and working together. Some examples of open source generative AI projects are:

• RunwayML: This helps people create new drawings, sounds, and videos with AI. You can change the code to customise the AI art! StyleGAN: This AI makes realistic fake human faces. The code is public so new things can be generated. Magenta: A tool from Google to make music with AI. You can build on it to train the AI in new ways. TensorFlow: This popular programming library lets anyone build all kinds of AI models. Generative AIs are created with it. Some other cool open source projects are GAN Lab, Mimetic, Lexica, and VQGAN. The code for all these is free to use and change! The benefit of open source is that anyone can learn from these projects, make them better, and create new innovations. You may be asking yourself, "Why should I choose open source generative AI over closed source?" The key advantage of open source generative AI is the collaborative and transparent nature of its development. With the code and models being public, many more developers can contribute, spot issues, suggest improvements, and build custom solutions suited to their needs. This pooling of knowledge and effort can often outpace what even large companies can do internally.

Understanding GPT-3

Have you ever wondered how your phone is able to predict the next word as you type? Or how can a website automatically translate into different languages? Behind the scenes, these apps use very complex artificial intelligence systems called neural networks. GPT-3 is one of the newest and most advanced neural networks ever made! GPT-3, or the Generative Pre-trained Transformer 3, is like a super-smart computer that is really good at understanding and generating human-like text. Created by OpenAI, think of it as a digital friend that can chat with you, write stories, answer questions, and do many other cool things with words.

A neural network is a type of AI algorithm designed to recognise patterns. It is inspired by the human brain. The 'neurons' are maths functions that connect in complex ways, similar to networks of real neurons in our heads. By analysing tons of data, the neural network learns to make predictions and decisions without needing explicit programming. This is the same way OpenAI's GPT-3 works.

GPT-3 has been trained on a massive amount of text from the internet. It's like having read billions and billions of books, articles, and conversations. This training allows it to understand the patterns and styles of human language. Now, when you chat with GPT-3, it uses its knowledge to generate responses.

You can ask it about history, science, or even tell it to create a story, and it will do its best to come up with interesting and relevant answers. It was trained on nearly half a trillion words from web pages, books, Wikipedia and all kinds

of online text! That's more text data than any other AI has ever learned from before. With this huge training dataset, GPT-3 gained an extremely advanced understanding of how human language works. It can generate sentences, have conversations, summarise long articles, translate languages, and more. GPT3 seems to truly grasp the nuances and complexity of human writing and speaking. Let's see an example. Say we give GPT-3 this prompt: 'Write a short children's story about a lost dog trying to find his way home'. Here is one possible story GPT-3 could generate in response: Rex the dog was wandering down the busy street, sad and alone. He had accidently chased a car and gotten lost from his family! He missed playing fetch with the kids and cuddling on the couch at night. Rex didn't know which way was home. He walked for a long time, asking people for help finding his house. But everyone was too busy to pay attention. Rex became hungry and tired. He napped on the sidewalk hoping his family would find him soon. Just as Rex was losing hope, he heard a familiar sound - the ice cream truck jingle! He remembered his home was always near where the ice cream truck parked. Rex followed the music all the way back to his street. His tail wagged wildly when he saw his house. The whole family shouted with joy and gave Rex a big hug when he came home. Rex was never lost again, and made sure to stay close when outside. But he knew if he got confused, he could listen for the ice cream truck to guide him. Rex was overjoyed to be back where he belonged.

Pretty impressive, right? The story has a clear beginning, middle and end. It has descriptive language, emotion, and fun kid-friendly details. GPT-3 learned all of this from studying so many books and articles written by humans. While not perfect, its writing ability is incredibly advanced for AI!

The possibilities with GPT-3 are endless. It can answer tricky questions, summarise long reports into bullet points, suggest code fixes for programmers, and more. Some even believe it's a step towards an AI that can truly reason and think like humans. Hugging Face's Transformers library has played a crucial role in making advanced language models, including GPT-3, accessible to developers. It fosters collaboration and accelerates innovation in the AI community.

Applications of GPT-3

Tools like GPT-3 are generally used for content creation, coding assistance, language translation, and chatbots. There are a few other applications of GPT-3 too.

Speech synthesis: GPT-3 can generate synthetic speech from text that sounds more natural, and has better intonation and emphasis.

Machine translation: Fine-tuned versions of GPT-3 significantly improve the quality of language translations compared to previous models.

Answering questions: It provides relevant and coherent answers to queries, drawing conclusions from textual evidence. This is helpful for search, education, and research.

Summarisation: It can distil the key points from long reports, research papers, articles and other documents into concise summaries.

Creativity and idea generation: GPT-3 can expand on prompts to generate ideas for stories, articles, ad campaigns, product names and other creative works.

GPT-3's limitations

While impressive in certain applications, GPT-3 has several shortcomings that are important to recognise. At first glance, GPT-3 seems amazingly capable - the hype makes it sound like a human-level AI. But under closer scrutiny, fundamental weaknesses become apparent. This is because it is only as good as the data it was trained on.

Accuracy issues: GPT-3 can produce answers that sound plausible but may not always be accurate. It doesn't have the ability to fact-check information, so its responses should be verified for correctness.

Inability to generate personal experiences: GPT-3 doesn't have personal experiences or emotions. If you ask it about its own feelings or experiences, it may generate responses based on general knowledge rather than personal understanding.
Redundancy: In certain situations, GPT-3 may provide more information than necessary or generate verbose responses. This can lead to outputs that sound plausible but are longer than needed.
No real-time learning: GPT-3 does not learn or adapt in real-time. Once it's trained, it remains static until the next training update. It doesn't learn from new information or experiences after its training data cut-off.
Building GPT-3 with Python
In the October 2023 issue of Open Source For You (page 72), I discussed how we can create our own ChatGPT bot in Telegram using an OpenAI API token. For your convenience I have attached my article link in the QR code box in Figure 1. Do go through that. We will now discuss building GPT-3 in the terminal. So, fire up your PC and start your favourite IDE. I will use Visual Studio Code, create a file named main. py and start the coding.
The prerequisites for this are:
OpenAI GPT-3 API key: You will need to sign up for access to the GPT-3 API on the OpenAI website (https://beta.openai.com/signup/). Once approved, you will get an
API key.
Python: Make sure you have Python installed on your system.
Building a simple chatbot with GPT-3
Install the required modules: Open a terminal or command prompt. Type pip3 install openai or (as per your system) pip install openai to install the
OpenAI Python library. You should get a confirmation message that the module has been successfully installed. Once this is done, you can type the command once again to verify if you have installed Python or not. You will get a message similar to the screenshot given in Figure 2. This means the OpenAI Python library has been installed on your system. This library provides a convenient way to interact with OpenAI's language models, including GPT-3. It simplifies the process of sending requests to the OpenAI API and handling responses.
Create a new file: Create a Python script; you can name the file anything you want (e.g., main.py) and use the OpenAI Python library to interact with GPT-3. Now let's begin with the coding.
First, obtain the API key and paste it in line 3. Initially, we have declared the function called 'generate_response', which will take a prompt as input and use the OpenAI API to generate a response based on the prompt. It returns the generated response as output.
The function code block is:
def generate_response(prompt): response = openai.Completion.create(engine="davinci-codex", prompt=prompt, max_tokens=150, n=1, stop=None
) return response.choices[0].text. strip()
Once we are done with this, it's time to collect user input as a prompt and conclude with output. To achieve this, we will declare another function called 'chat_with_bot()' and work with a while loop to make the condition work in a loop, once the result is shared with the user. To do that, we declare a loop with the syntax of 'while True:' and inside the 'chat_with_bot()' function is a Python function that implements a simple chatbot. It takes user input, generates a response using the OpenAI Codex API, and prints the response to the console. The chatbot continues to interact with the user until the user enters 'exit'.
The function starts by printing a greeting message to the console. It enters a while loop that continues until the user enters 'exit'. Inside the loop, it prompts the user for input using the input function. If the user enters 'exit', the function prints a goodbye message and breaks out of the loop. Otherwise, it constructs a prompt by appending the user input to the string 'User:' and assigns it to the variable 'prompt'. It calls the 'generate_response' function, passing the prompt as an argument, to generate a response using the OpenAI Codex API. The response is extracted from the API response object and assigned to the variable response. The response is printed to the console with the prefix 'Chatbot:'.
The function code block is:
def chat_with_bot():
print("Chatbot: Hi there! Ask me anything or just chat with me.") while True: user_input = input("You: ") if user_input.lower() == 'exit': print("Chatbot: Goodbye!") break prompt = f"User: {user_input}\ nChatbot:" response = generate_ response(prompt)
print(f"Chatbot: {response}")

To sum up, your complete code block should look somewhat similar to the code screenshot given in Figure 3, and the code should not show any syntax error. If everything works well, we are ready to fetch the response from the chatbot. So we will run this bot by typing the command 'python3 main.py' (sometimes 'python bot.py' or 'py bot.py' works as well). Hence, please make sure Python is available on your system. If you have done everything correctly, you will get an output of a greeting message. Please remember this is just a basic example of how you can connect OpenAI's GPT-3 code to build a chatbot.

Important notes

Replace 'YOUR_API_KEY' with the actual API key you received from OpenAI.

Experiment with different prompts, engines, and parameters to achieve the desired chatbot behaviour. Be mindful of OpenAI's usecase policies and ethical guidelines when creating and deploying chatbots.

Keep in mind that developments are continuous; so it's a good idea to refer to the latest OpenAI documentation for any changes or additional features.

The power of generative AI, exemplified by GPT-3, has far-reaching implications for technology, business, and society. As we look forward to future iterations, collaborative efforts and ethical considerations will shape the transformative impact of generative AI on our world.

The author specialises in RPA, Python, and Web 3.0, and has over five years of expertise in creating and managing Telegram bots.