ChatGPT by OpenAI

What is ChatGPT?

ChatGPT is an advanced AI language model developed by OpenAI, leveraging the GPT-4 architecture. It is designed to understand and generate human-like text based on the input it receives, enabling it to engage in conversations, answer questions, provide explanations, and assist with various language-related tasks. Trained on a vast corpus of diverse internet text, ChatGPT learns patterns, structures, and facts about language, allowing it to perform tasks such as writing emails, essays, or code, and summarizing information. However, it has limitations, including the potential for generating incorrect or nonsensical answers, lacking real-time information access, and occasionally producing biased or inappropriate content. Despite these limitations, ChatGPT is a powerful tool used in applications like customer service chatbots, educational tools, content creation, and as an assistant in software development and data analysis, significantly enhancing productivity and providing valuable assistance in numerous domains.

Overview of Functionality:

Core Functionality:

1. Natural Language Understanding (NLU):

- Contextual Understanding: ChatGPT can interpret and maintain context over multiple turns of conversation. It uses its understanding of previous inputs to generate coherent and contextually appropriate responses.
- Language Comprehension: It processes and comprehends a wide range of topics and language nuances, enabling it to provide relevant answers and engage in meaningful conversations.

2. Text Generation:

- Coherent Responses: ChatGPT generates text that is coherent and fluent, mimicking human language patterns. It uses its training on vast amounts of text data to produce responses that are contextually relevant.
- Creative Writing: Beyond straightforward responses, it can generate creative and novel text, such as stories, poems, or explanations, based on the input and prompts it receives.

3. Conversational Agent Capabilities:

- Chatbot Functionality: It serves as the backbone for creating conversational agents and chatbots that can interact with users in natural language. This includes customer support bots, virtual assistants, and interactive tools.
- Multi-turn Dialogues: It supports ongoing conversations where it remembers previous exchanges and maintains consistency in its responses over the course of a conversation.

4. Adaptability and Customization:

- Fine-tuning: Developers can fine-tune ChatGPT for specific tasks or domains by providing additional training data. This improves its performance in specialized applications such as technical support, medical advice, or legal consultations.
- API Integration: It can be integrated into various applications via APIs, allowing developers to leverage its capabilities within their own software environments.

5. Ethical Considerations and Safety:

- Bias Mitigation: OpenAI works on mitigating biases in ChatGPT's training data and output to ensure fairness and inclusivity in its responses.
- Safety Protocols: It includes safety protocols to prevent the generation of harmful or inappropriate content, helping maintain a positive user experience and ethical standards.

6. Scalability and Performance:

- Large-scale Deployment: ChatGPT's architecture allows it to scale effectively, supporting large user bases and handling simultaneous interactions across multiple platforms.
- Performance Improvements: With each iteration (like GPT-3.5), OpenAI continues to improve ChatGPT's performance in terms of response quality, speed, and efficiency.

Use Cases:

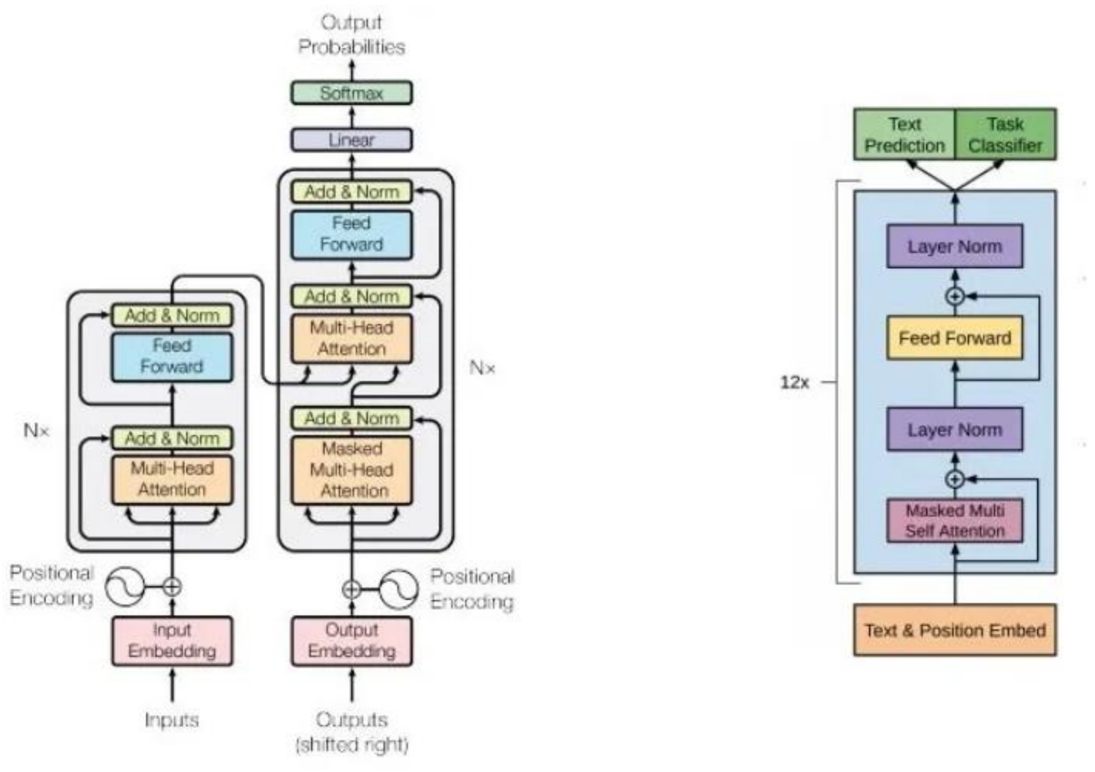
1. Customer Support: Providing automated responses and handling inquiries in various industries.
2. Education: Supporting tutoring and learning platforms by answering questions and explaining concepts.
3. Content Generation: Assisting in writing articles, generating product descriptions, or creating marketing materials.

Technical Details :

High-Level Architecture of ChatGPT:

GPT 3 Language Model

GPT-3 (Generative Pre-trained Transformer 3) is a language model that was created by OpenAI. The 175-billion parameter deep learning model is capable of producing human-like text and was trained on large text datasets with hundreds of billions of words.



Transformer GPT

GPT uses an unmodified Transformer decoder, except that it lacks the encoder attention part. We can see this visually in the above diagrams. The GPT, GPT2, GPT 3 is built using transformer decoder blocks. BERT, on the other hand, uses transformer encoder blocks. GPT-3 was trained with huge Internet text datasets — 570GB in total. When it was released, it was the largest neural network with 175 billion parameters (100x GPT-2).GPT-3 has 96 attention blocks that each contain 96 attention heads.

Transformers Post .GPT-3 was trained using "next-word prediction, task a kind of Unsupervised training in which it predicts the next word in a sentence. The input sequence is actually fixed to 2048 words (for GPT-3). We can still pass short sequences as input: we simply fill all extra positions with "empty" values. The GPT output is not just a single guess, it's a sequence (length 2048) of guesses (a probability for each likely word). One for each 'next' position in the sequence. But when generating text, we typically only look at the guess for the last word of the sequence.

Encoding-:But wait a second, GPT can't actually understand words. The first step is to keep a vocabulary of all words, which allows us to give each word a value. yashu is 0, shobham is 1, and so on. (GPT has a vocabulary of 50257 words).It uses sub words algorithms to create the vocab, if word is not present in the current dictionary of model, GPT-3 actually uses byte-level Byte Pair Encoding (BPE) sub word tokenization method .Based on sub words BPE algo it creates positional encodings where it convert sub words to the vectors..

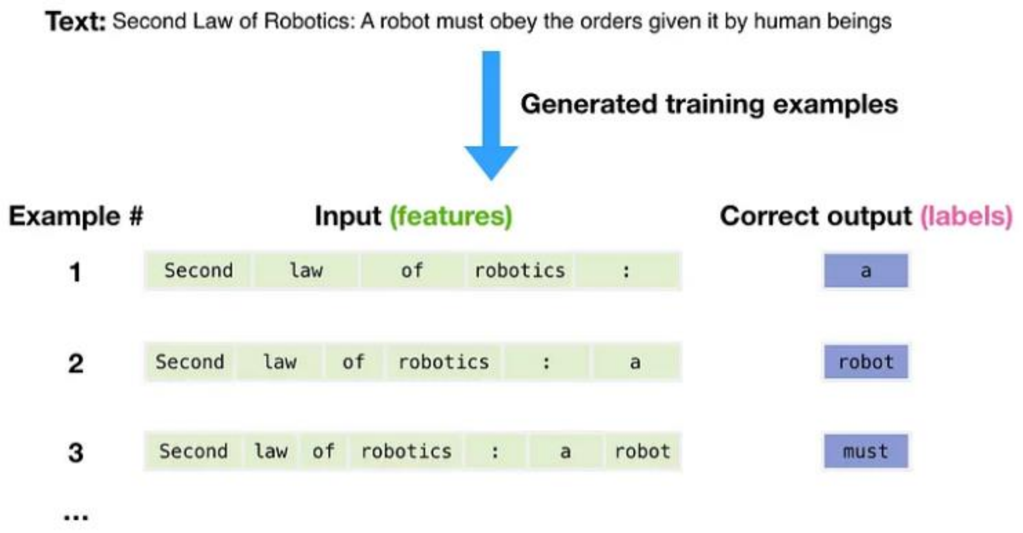
Multi Head Attention-:

1. Once Encoding is done , We calculate the key, query, Value Vectors for each token of our sequence using dot product matrix multiplication to produce a score matrix. The score matrix determines how much focus should a word be put on other words. So each word will have a score that corresponds to other words in the time-step. The higher the score the more focus (attention) will be there.

2. Then, the scores get scaled down by getting divided by the square root of the dimension of query and key. This allows more stable gradients, as multiplying values can have exploding effects.Then we do the SoftMax of scaled score to get the attention weights, which gives probability values between 0 and 1.This allows the model to be more confident about which words to attend too

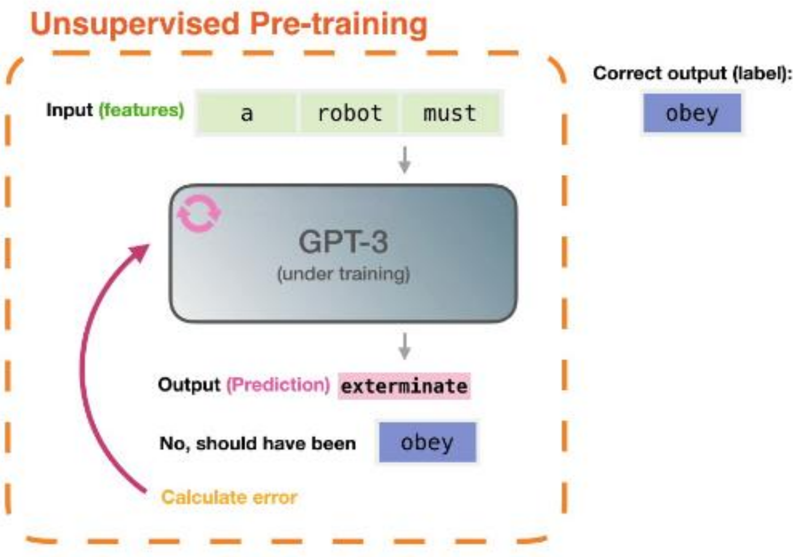
GPT-3 Training

The dataset of 300 billion tokens of text is used to generate training examples for the model. For example, these are three training examples generated from the one sentence.



Text: Second Law of Robotics: A robot must obey the orders given it by human beings
Generated training examples
Example #
Input (features)
Correct output (labels)
1
Second law of robotics :
a
2
Second law of robotics : a
robot
3
Second law of robotics : a robot
must
...

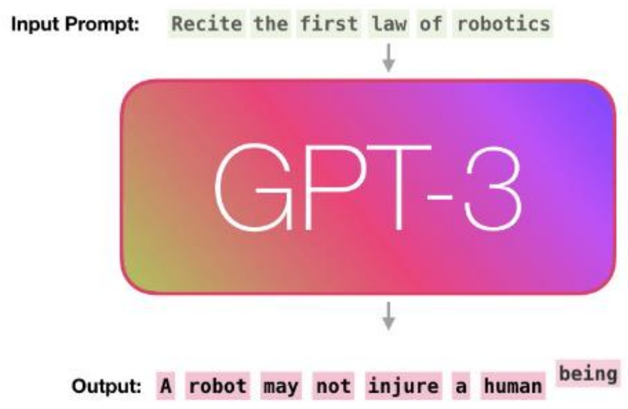
Since Pretraining objective is Next word Prediction. The model is presented with an example. We only show the features and it will predict the next word.



GPT-3 Pretraining

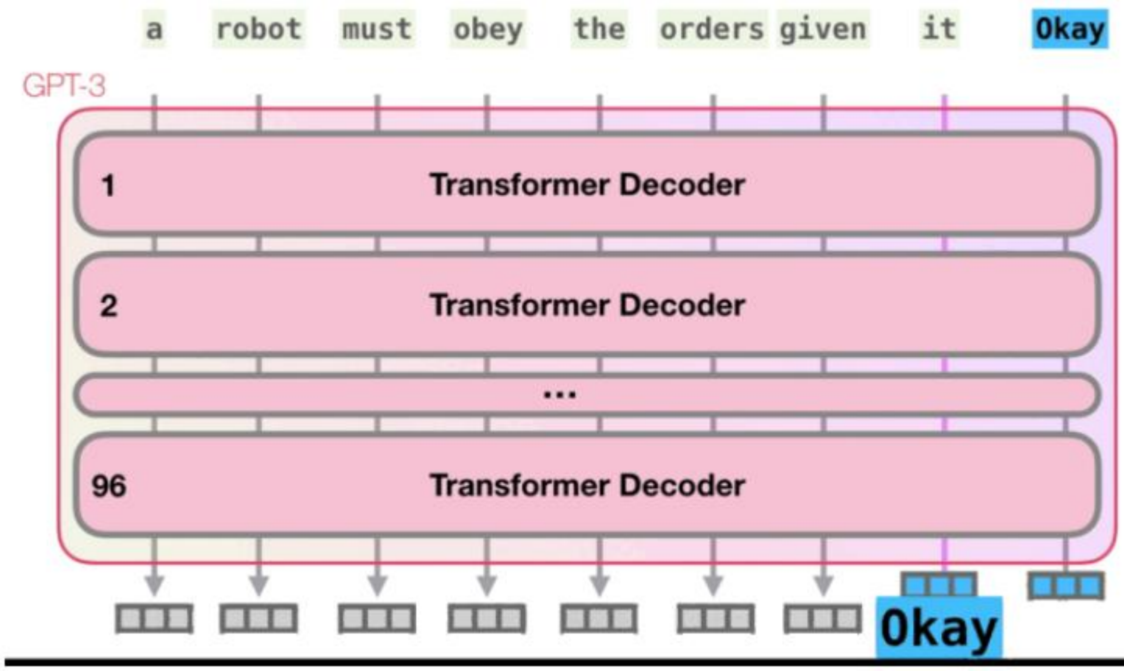
GPT-3 Pretraining

we are passing tokens to the model .Model is predicting one token at a time since it is auto regressive model. First time the model's prediction will be wrong. We calculate the error by showing the correct output and update the model parameters, so next time it makes a better prediction. This process will be repeated many times.


Input Prompt:
Recite the first law of robotics
GPT-3
Output:
A robot may not injure a human
being

es.

The untrained model starts with random parameters. Training finds values that lead to better predictions. Every time it outputs a token. It will be added to the Input sequence to predict the next token until it finds a END token. The future tokens are masked in the Input sequence while training and are shifted right by1 token. Below is the detailed intuition



Above GIF represents the input and response (“Okay human”) within GPT3. Notice how every token flows through the entire layer stack. We don’t care about the output of the first words. When the input is done, we start caring about the output. We feed every word back into the model.This is how GPT-3 works .

CHATGPT- Explained step by step!!

It is a variant of the popular GPT-3 (Generative Pertained Transformer 3) model discussed above, which has been trained on a massive amount of text data to generate human-like responses to a given input. Chat GPT was modified and improved using both supervised and reinforcement learning methods, with the assistance of human trainer (RLHF).Chat GPT also has 176 billion parameters same as GPT -3 model. The learning includes 3 Steps.

1. Supervised fine tuning of GPT 3.5 Model
2. Reward Model
3. Proximal Policy Optimization (PPO)

Supervised fine tuning (Step1)

In first Step a pretrained GPT-3 model is used and it will be fine tuned with the help of labelers by creating a supervised dataset. Input Queries were collected from the actual user entries and model generated different responses with respect to that input prompts. The labelers then wrote an appropriate response to the input prompt’s (how they want to see that prompt to be answered).The GPT-3 model was then fine-tuned using this new supervised dataset, to create GPT-3.5 model.

Reward Model (Step 2)

After the SFT 3.5 model is trained in step 1, the model generates better responses to input prompts. In this step SFT model is used and different input/prompts queries fed to the finetuned model and different responses were generated (4 to 7)for every input/prompt. Then labeler determines a reward for each of these outcomes and this reward is proportional to the quality of response with respect to initial prompt .The Labeler rank the output's in sequence order of best to worst. Then we can use this data in order to train a reward model . The input for a reward model will be the user prompt and one of the responses we generated and output will be a scaler value which determines the quality of response with respect to the input prompt. Also we use rankings that we generated in the past in order to train this reward model.

Proximal Policy Optimization (PPO) RL algo- Step3

In this step we pass unseen input sequences to the clone SFT model we got in step1. The model will generate response with respect to the input prompt. We pass the response to our reward model which we got in step 2 to understand, how high quality was this response for that input prompt and the output reward will be used to finetune the parameters of our SFT model .This is how our SFT model will incorporate more human like characteristics and behavior’s via Reinforcement Learning.

To detail the key API endpoints, request/response formats, and functionality of the ChatGPT OpenAI API, here's a breakdown based on typical functionalities:

1. Completion API Endpoint

Endpoint: `/v1/engines/{engine}/completions`

Request:

- Method: POST
- Headers:
    1. Authorization: Bearer YOUR_API_KEY`

    2.`Content-Type: application/json`

- Body:

```json
{
  "prompt": "Your prompt text here",
  "max_tokens": 50,
  "temperature": 0.7,
  "top_p": 1.0
```

}

~~~

- `prompt`: The text you want to generate completion for.

- `max_tokens`: Maximum number of tokens in the generated completion.

- `temperature`: Controls randomness in generation (0.0 for deterministic, higher for more randomness).

- `top_p`: Nucleus sampling parameter for diversity (1.0 means no truncation).

Response:

- Status Code: 200 OK

- Body:

```json
{
  "id": "string",
  "model": "string",
  "choices": [
    {
      "text": "Generated text here",
~~~

```
        "index": 0,
        "logprobs": {
          "tokens": {
            "text": "Token probability pairs"
          }
        },
        "finish_reason": "string"
      }
    ]
  }
```

- `text`: Generated completion text.

- `logprobs`: Log probabilities for tokens generated.

- `finish_reason`: Reason for completion (e.g., "length", "stop").

2. Search API Endpoint

Endpoint: `/v1/engines/{engine}/search`

Request:

- Method: POST

- Headers:
  - `Authorization: Bearer YOUR_API_KEY`
  - `Content-Type: application/json`
- Body:

```json
{
  "documents": [
    {"text": "Document 1 text"},
    {"text": "Document 2 text"}
  ],
  "query": "Your search query",
  "file": "Optional file URL for context"
}
```

  - `documents`: List of documents for search context.
  - `query`: Search query text.
  - `file`: Optional file URL for additional context.

Response:

- Status Code: 200 OK
- Body:

```json
{
  "results": [
    {
      "document": {
        "text": "Matched document text",
        "metadata": {}
      },
      "score": 0.95
    }
  ]
}
```

- `results`: List of matched documents with scores.

- `score`: Relevance score of the matched document.

Step 2 The reward model