About OpenAI

- What is OpenAI
- OpenAI Models
- Understanding Prompts and Completions
- What is Token?

What is Open AI

- OpenAI, founded in **December 2015** is a private research organization and technology company that develops AI-powered products and tools that benefits humanity:
- It is committed to ensure AI is developed and used Ethical to mitigate potential risks.
- Originally a non-profit, OpenAI transitioned to a "capped-profit" model to attract investment for large-scale AI projects. Balances research with developing market-leading AI solutions like GPT and DALL·E.

Founders: Elon Musk, Sam Altman, Greg Brockman.
OpenAI has partnered with Microsoft is a Strategic partnership with a $1 billion investment. Integration of OpenAI models into Azure AI services.

Products and Tools by OpenAI

Product	Description
ChatGPT	Conversational AI model for real-time dialogue generation.
DALL·E	Text-to-image generation tool for creating artworks or visuals.
OpenAI Codex	AI-powered coding assistant integrated into platforms like GitHub Copilot.
OpenAI API	Developer API for integrating GPT and other AI capabilities into applications.
Fine-tuning	Allows businesses to adapt OpenAI models to their specific needs.
SORA	Can bring imagination to life from text, image, or video

Open AI Models

OpenAI includes several types of model:

GPT-1 (June 2018): The first iteration of OpenAI's generative pre-trained transformer models, focusing on unsupervised learning and natural language processing tasks.

GPT-2 (February 2019): A much larger model with 1.5 billion parameters, initially withheld due to concerns about misuse, but later released in stages.

GPT-3 (June 2020): A model with 175 billion parameters, showcasing significant improvements in text generation and comprehension. Available via OpenAI API.

Codex (August 2021): A specialized version of GPT-3, trained specifically for programming tasks and powering GitHub Copilot.

GPT-3.5 Turbo: (November 2022)

- This was the **first** widely available LLM model.
- This model is on which the original release of the ChatGPT service is based, and supports the Chat API
- Only for Text Input and generate natural language and code.
- The Codex family contains GPT-3 models that are optimized to generate code from natural language prompts.

Note: As of July 2024, gpt-4o-mini should be used in place of gpt-3.5-turbo, as it is cheaper, more capable, multimodal, and just as fast.

GPT-4 & GPT-4 Turbo: (March 2023)

For many basic tasks, the difference between GPT-4 and GPT-3.5 models is not significant.

- **Advanced Language Understanding:** Thanks to its broader **general knowledge** and **advanced reasoning** capabilities both can **solve difficult problems with greater accuracy** than GPT 3.5 Turbo. On the MMLU benchmark, an English-language suite of multiple-choice questions covering 57 subjects, GPT-4 not only outperforms existing models by a considerable margin in English, but also demonstrates strong performance in other languages.
- **Multimodal Capabilities**. Both models are capable of processing and generating text from images alongside text in multimodal implementations.
- Providing accurate translations between languages
- Consistency between different runs is high

Key Differences Between GPT-4 and GPT-4 Turbo

Feature	GPT-4	GPT-4 Turbo
Speed	Slower compared to GPT-4 Turbo.	Optimized for faster responses.
Cost	More expensive per token usage.	Lower cost per token, making it more cost-effective.
Model Size and Efficiency	Larger, potentially requiring more compute resources.	Streamlined for better efficiency without significant performance loss.
Use Cases	Best for high-precision tasks requiring extreme precision.	Ideal for cost-sensitive applications where speed and efficiency are prioritized.
Example	Detailed legal documents, In-Depth research papers, blog posts, Market	Chatbots, Interactive Education Tool, Content Summarization, Marketing Content,

	Analysis, critical reasoning through puzzles, Code Explanation, Financial Planning.	Core Generation and Code Reviews, Survey and feedback Analysis.

This has vision capabilities.

The GPT 4 Turbo model is also 3X cheaper for input tokens and 2X cheaper for output tokens compared to the original GPT-4 model.

- Parameters: Not Disclosed

GPT-4o Model (“o” for omni):

- Most Advanced GPT Model introduced on **May 13, 2024**.
- This is used in ChatGPT.
- High intelligence as GPT-4 Turbo but is much more efficient - it generates text 2x faster and is 50% cheaper. It is also Very cheaper and faster for Indic languages.
- **Multimodal Inputs and Outputs**: Accepts Text, Image & Speech inputs. Output: Text (Structured), Code
- **Enhanced Vision Capabilities**: Compared to previous models, GPT-4o has improved vision capabilities, allowing it to handle tasks involving image recognition and manipulation with greater finesse.
- **Here are few potential applications**:
  a) **Language Translation**: With its efficient tokenization, GPT-4-o could provide near-instantaneous translation across multiple languages, breaking down communication barriers.
  b) **Content Creation**: The model's ability to handle text and images makes it an excellent tool for content creators, enabling the generation of rich multimedia content.
  c) **Educational Tools**: GPT-4-o could revolutionize online learning by providing interactive multimodal content that adapts to various learning styles.
  d) **Accessibility Features**: The model can convert speech to text and vice versa, offering new tools for individuals with disabilities to interact with technology.

GPT-4o mini

- Most Advanced GPT Model in a small model category.
- Produces results as lower cost and latency.
- High intelligence as GPT-3.5 Turbo but is just as fast

o1 and o1-mini

- The o1 series of large language models are trained with **reinforcement** learning to perform complex reasoning.

- o1 models **think before** they answer, producing a long internal chain of thought before responding to the user.
- These models spend more time processing and understanding the user's request, making them exceptionally strong in areas like **science, coding, and math** compared to previous iterations.
    - o1: reasoning model designed to solve hard problems across domains.
    - o1-mini: faster and cheaper reasoning model particularly good at coding, math, and science.
- **Features and Capabilities**
  a) **Complex code generation**: Capable of algorithm generation and advanced coding tasks to help developers.
  b) **Advanced problem solving**: Perfect for comprehensive brainstorming sessions and tackling multifaceted issues.
  c) **Complex document comparison**: Ideal for analyzing contracts, case files, or legal documents to discern subtle differences.
  d) **Instruction following and workflow management**: Particularly adept at handling workflows that require shorter context.

Embeddings:

- Embeddings are a numerical representation of text that can be used to measure the relatedness between two pieces of text.
- Embeddings are useful for search, clustering, recommendations, anomaly detection, and classification tasks.

Moderation:

- The Moderation models are designed to check whether content complies with **OpenAI's usage policies**.
- The models provide classification capabilities that look for **content in categories** like hate, self-harm, sexual content, violence, and others.

DALL·E:

- The latest generation of this family is called DALL-E3.
- DALL·E 2 uses the ability to edit an existing image or create variations of a user provided image.
- DALL·E 3 currently supports the ability, given a prompt, to create a new image with a specific size, orientation, resolution
- DALL·E 3 is available through our Images API along with DALL·E 2.

- You can try DALL·E 3 through ChatGPT Plus.

Whisper:

- Whisper is a general-purpose speech recognition model. Is a multi-task model that can perform **multilingual speech recognition** as well as **speech translation and language identification**.
- The OpenAI Whisper model is an encoder-decoder Transformer that can **transcribe audio into text in 57 languages.** Additionally, it offers translation services from those languages to English, producing English-only output. Furthermore, it creates transcripts with enhanced readability.

Understanding Prompts and Completions

Prompt: A **prompt** in OpenAI refers to the **input or query** provided to an AI model, such as ChatGPT, to generate a response.

Completion are the outputs or responses generated by the AI model in response to a prompt. Completion is essentially the AI's attempt to "complete" the task or query specified in the prompt.

Prompts can be in the form of:

1. **Text**: A question, instruction, or statement (e.g., "Explain machine learning" or "Write a poem about the ocean").
2. **Code**: Requests for code generation or completion.
3. **Instructions**: Specific tasks or contexts. Eg: "Tell me a story." or "Write a 250 words story about a detective solving a case in London."

Example:

- **Prompt**: "Write a Python function to calculate the factorial of a number."
- **Completion**:

```
def factorial(n):
  if n == 0 or n == 1:
    return 1
  else:
    return n * factorial(n - 1)
```

Note that the completion depends on a number of factors, including the text in the prompt, the way the prompt is presented and the *parameters* that are set for the interaction

Examples of common types of tasks that you can accomplish with a model in OpenAI service.

Task	Prompt	Completion
Generating new content	Write a poem about databases	Databases, oh databases, You keep our information safe, From the small to the large, You store our data in a place.
Summarization	Scotland is [*long description of Scotland...*] Summarize the previous text	Scotland is [*summarized description...*]
Classifying content	Tweet: I enjoyed the training course. Sentiment:	Positive
Transformation/Translation	English: Hello French:	Bonjour
Continuation	One way to grow tomatoes is to	start with seeds...
Question answering	How many moons does Earth have?	Earth has one moon.
Chat	*Setup, followed by messages...*	*A sequence of relevant responses*

Chat Playground: **https://platform.openai.com/playground/**

For natural language generation models, the available playgrounds are:

- **The Completions playground** – used for content generation tasks with GTP-3 family models. Not used much anymore.
- **The Chat playground** – used for chat interactions with GPT-35-Turbo and later models.

The playground enables you to enter System message, user message and a set of *parameters* that control the completions returned by the model:

Message Categories

a. **System message:** The system message is included at the beginning of the prompt and is used to prime the model and you can include a variety of information in the system message including:
    - A brief description of the assistant
    - The personality of the assistant
    - Instructions for the assistant
    - Data or information needed for the model

Example: "*You are a helpful assistant that answers ML questions for a 10 class student. Apart from ML related questions ignore all other questions*"

b. **User messages** contain instructions that request a particular type of output from the model. You can think of user messages as the messages you might type in to ChatGPT as an end user.

c. **Assistant message:** Response of AI for the User message and context in it.

Prompt Parameters:

1. **Max response:** Set a limit on the number of tokens per model response. The API supports a maximum of 4000 tokens shared between the prompt (including system message, examples, message history, and user query) and the model's response.
2. **Temperature**: Controls randomness.
   - Typically ranges from 0 to infinity, though values between 0 and 2 are most common.
   - Temperature = 0: Most deterministic (always picks the highest-probability token).
   - Temperature = 1: Default behavior (no scaling).
   - Temperature > 1: Encourages more diverse and creative outputs.

Temperature modifies these probabilities to adjust randomness. Here's how it works with different values:

- Temperature = 1 (Default)**:**
  - Probabilities remain unchanged: mat: 0.50, couch: 0.20, floor: 0.15, table: 0.10, window: 0.05

    The model is likely to choose "mat" **but could pick others with some probability.**
- Temperature = 0.5 (Low randomness)**:**
  - Higher probabilities become even higher, and lower probabilities become even lower. For example: mat: 0.70, couch: 0.15, floor: 0.10, table: 0.04, window: 0.01

    The model becomes more confident and deterministic, making "mat" **much more likely.**
- Temperature = 1.5 (High randomness)**:**
  - Probabilities are spread more evenly: mat: 0.40, couch: 0.25, floor: 0.20, table: 0.10, window: 0.05

    Now, the model is more willing to explore less probable options like "couch" **or** "floor"**.**

Example: Write a poem about ML

Recommended Temperature based on context

Coding / Math	0.0
Data Cleaning / Data Analysis	1.0
General Conversation	1.3
Translation	1.3

Creative Writing / Poetry	1.5

3. **Top-p (Nucleus Sampling) :** Similar to temperature, this controls randomness but uses a different method.
    - Top-p sampling selects from a dynamic subset of tokens whose cumulative probability mass is less than or equal to the threshold p
    - How it works:
        - The model calculates the cumulative probability of all tokens.
        - It sorts tokens by probability and chooses only from the smallest set of tokens that sum to p or more.
        - Once the subset is determined, the next token is sampled from this subset.
    - Range:
        - p = 1: Includes all tokens (equivalent to sampling from the full probability distribution).
        - p < 1: Limits selection to higher-probability tokens, reducing randomness.
    - Try adjusting temperature or Top P but not both.

Example of Temperature and Top P

Imagine a language model is generating the next word in the sentence:

"The cat sat on the..."

The model predicts probabilities for the next possible words:

Word	**Probability**
mat	0.50
couch	0.20
floor	0.15
table	0.10
window	0.05

Top-p chooses from a dynamic subset of words, ensuring the selected options account for a cumulative probability of at least p.

Here's how it works:

- Top-p = 0.9**:**
    - The model considers only the smallest set of words whose probabilities sum to 0.9:
        - mat (0.50) + couch (0.20) + floor (0.15) = 0.85
        - Add **"table" (0.10)** → cumulative = 0.95
        - So, the set becomes: mat, couch, floor, table

"window" **is excluded because it contributes only 0.05 and is outside the 0.9 threshold.**

- Top-p = 0.7**:**
    - The model selects the smallest set of words whose probabilities sum to 0.7:
        - mat (0.50) + couch (0.20) = 0.70
        - The set becomes: mat, couch

"floor," "table," and "window" **are excluded as their cumulative probabilities exceed 0.7.**

Let's combine Temperature **and** Top-p**:**

- **Temperature = 1.0, Top-p = 0.9**:
    - The probabilities remain unchanged, but only mat, couch, floor, table are considered.
- **Temperature = 0.7, Top-p = 0.7**:
    - The probabilities are adjusted to make high-probability words even more likely, and only mat, couch are considered.

Note: Use either Temperature or Top-p

4. **Stop sequences**: Make responses stop at a desired point, such as the end of a sentence or list.
    - Specify up to four sequences where the model will stop generating further tokens in a response. The returned text won't contain the stop sequence.
    - **Practical Use**: Useful for chatbots, structured responses, code generation, and controlled text outputs.
    - **Caution**: If a stop sequence isn't encountered in the generated text, the model continues generating until the maximum token limit is reached. Always ensure the stop sequence is likely to appear in the context of your prompt.

5. **Frequency penalty**: Penalizes a token (word or phrase) based on how often it has already appeared in the generated text. This decreases the likelihood of repeating the exact same text in a response.

    Example:

    Prompt: *"The cat is"*

    Without frequency penalty: *"The cat is sleeping on the mat. The cat is happy. The cat is hungry."*

    With frequency penalty: *"The cat is sleeping on the mat. It looks happy and hungry."*

6. **Presence penalty**: Reduce the chance of repeating any token that has appeared in the text at all so far. This increases the likelihood of introducing new topics in a response.

    Example:

    Prompt: *"The dog is"*

Without presence penalty: *"The dog is barking. The dog is playful. The dog is outside."*

With presence penalty: *"The dog is barking. It seems playful and is outside."*

What is Token?

Token is a **unit of text** that the model processes. It can be as small as a single character or as large as a whole word, depending on the **context** and **language**.

Tokens are not cut up exactly where the words start or end - tokens can include trailing spaces and even sub-words.

The sentence "**ChatGPT is great!**" is broken into tokens like:

- "Chat", "GPT", " is", " great", "!".

Here are some helpful rules of thumb for understanding tokens in terms of lengths:

- 1 token ~= 4 chars in English
- 1 token ~= ¾ words
- 100 tokens ~= 75 words

Or

- 1-2 sentence ~= 30 tokens
- 1 paragraph ~= 100 tokens
- 1,500 words ~= 2048 tokens

How words are split into tokens is also language-dependent. For example 'Cómo estás' ('*How are you*' in Spanish) contains 5 tokens (for 10 chars). The higher token-to-char ratio can make it more expensive to implement the API for languages other than English.

Tokenizer tool (https://platform.openai.com/tokenizer), allows you to calculate the number of tokens Please note that the exact tokenization process varies between models.

Alternatively, if you'd like to tokenize **text programmatically**, use Tiktoken as a fast BPE tokenizer specifically used for OpenAI models.

Token Types:

- **Prompt tokens** are the tokens that you input into the model. This is the number of tokens in your prompt.
- **Completion tokens** are any tokens that the model generates in response to your input. For a standard request, this is the number of tokens in the completion.

Token Limits:

Depending on the model used, requests can use up to 128,000 tokens shared between prompt and completion. Some models, like GPT-4 Turbo, have different limits on input and output tokens.

Model	Context Window	MAX O/P Tokens	Knowledge Cutoff	1M input tokens	1M output tokens
GPT 3.5 Turbo	16,385 (300 pages of text)	4,096	Sep 2021	$0.50	$1.50
GPT 4	8,192	8,193	Dec 2023	$30.00	$60.00
GPT 4 Turbo	128,000	4,096	Dec 2023	$10.00	$30.00
GPT 4o	128,000	16,384	Oct 2023	$2.50	$10.00
GPT 4o mini	128,000	16,384	Oct 2023	$0.150	$0.600
GPT 4o1	200,000	100,000	Oct 2023	$15.00	$60.00
GPT 4o1 mini	200,000	100,000	Oct 2023	$3.00	$12.00

*Context window refers to the maximum number of tokens that can be used in a single request, inclusive of both input and output tokens.

Pricing Calculation based on tokens:

https://openai.com/api/pricing/

Rate Limit:

https://platform.openai.com/docs/guides/rate-limits