Mastering Prompt Engineering

Xavier Morera

Helping developers create epic applications

www.xaviermorera.com / www.bigdatainc.org / www.lupo.ai

Google

what is an llm

All Images Videos Books News More Tools SafeSearch

About 250,000,000 results (0.42 seconds)

TechTarget
https://www.techtarget.com › large-language-model-LLM

What are Large Language Models? | Definition from ...

A large language model (LLM) is **a type of artificial intelligence (AI) algorithm that uses deep learning techniques and** massively large data sets to ...

Deep learning · How do Big Data and AI Work... · Transformer neural networks

People also ask

What is the meaning of LLM?

What is an LLM in AI?

What is the difference between LLM and AI?

What are LLMs examples?

Feedback

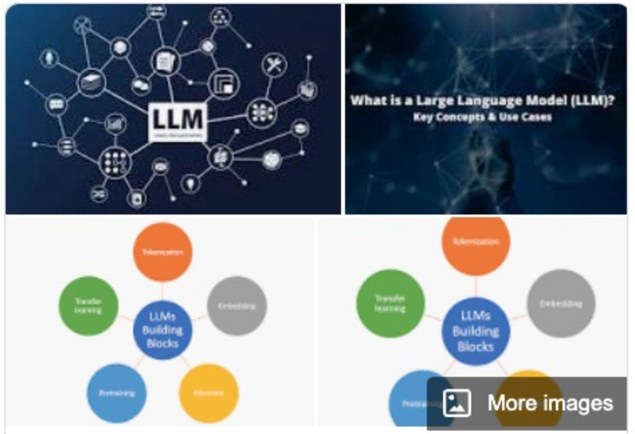
Computerworld
https://www.computerworld.com › article › what-are-l...

What are LLMs, and how are they used in generative AI?

May 30, 2023 — An LLM is **a machine-learning neuro network trained through data input/output sets**; frequently, the text is unlabeled or uncategorized, and the ...

Wikipedia
https://en.wikipedia.org › wiki › Large_language_model

Large language model



Large language model

A large language model is a type of language model notable for its ability to achieve general-purpose language understanding and generation. Wikipedia

Feedback

What is an LLM (Large Language Model)?

An LLM, or Large Language Model, is a type of artificial intelligence system that has been trained on a massive amount of text data to understand and generate human-like text. These models are designed to process and generate natural language, making them particularly adept at tasks such as text generation, text completion, translation, text summarization, and more. LLMs are based on deep learning techniques, specifically neural networks, and they are capable of understanding and generating text in multiple languages.

One of the most well-known examples of an LLM is GPT-3 (Generative Pre-trained Transformer 3), which was developed by OpenAI. GPT-3 has 175 billion parameters and can generate coherent and contextually relevant text based on the input it receives.

LLMs have a wide range of applications, including natural language understanding, chatbots, virtual assistants, content generation, language translation, sentiment analysis, and more. They have significantly advanced the state of the art in natural language processing and have the potential to revolutionize various industries and applications by automating or enhancing language-related tasks.

Regenerate

Send a message

Understanding Prompting

Prompt

Input provided by users or programmers to get a response from an LLM

Prompt

Input provided by users or programmers to get a response from an LLM

Establishes the context and offers instructions that guilde the LLM to create a relevant and meaningful response

Prompt Sample

"Write a story on a traveling computer programmer"

Prompt

Natural language

How we speak in real life

Prompts bridge the gap between human intention and machine-generated responses

Crafting effective prompts is pivotal in ensuring that an LLM produces accurate and valuable responses

Crafting Effective Prompts

Good	Better, Best, Way Better...
Write about technology.	Write an engaging blog post about the impact of artificial intelligence on everyday life, highlighting both its benefits and potential concerns.

Anatomy of an Effective Prompt

Anatomy of an Effective Prompt

Clarity

Specificity

Contextual information

Desired format

Length constraints

Examples or hints

Clarity

A clear prompt is easy for both humans and AI models to understand

It should convey your request or question without ambiguity

Specificity

An effective prompt is specific and well-defined

It should clearly state the information or data you want from the AI model

Contextual Information

Including relevant context can help the AI model generate more accurate responses

Provide background information or constraints, if necessary

Desired Format

Specify the format you want the answer in

It can be a list, paragraph, chart, or any other format that suits your needs

Length Constraints

Specify the desired length, as a guide for the AI model

Examples or Hints

Providing examples or hints can clarify your request and guide the AI model

An invaluable rule when creating prompts!

Avoid bias or controversy

An invaluable rule when creating prompts!

Avoid Bias or Controversy

Avoid Bias or Controversy

- Use neutral and inclusive language
- Avoid framing questions in a way that assumes or promotes stereotypes or discrimination
- Ensure prompts do not introduce bias
- Consider other perspectives
- Test prompts with a diverse group of people to identify and address potential issues

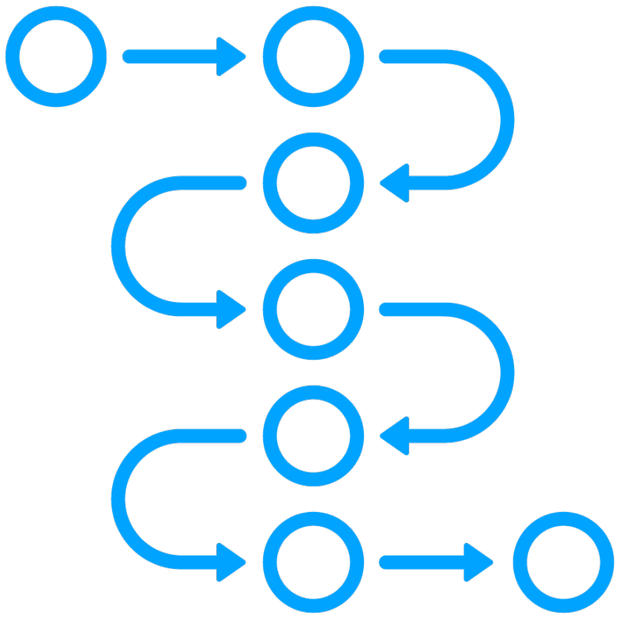
Multiple Prompts for Complex Tasks

For complex tasks

- Consider breaking down your prompt

Multiple prompts may explain the scenario in a much better way

Iteration and Refinement



Iterate and refine prompts

Experimentation is key

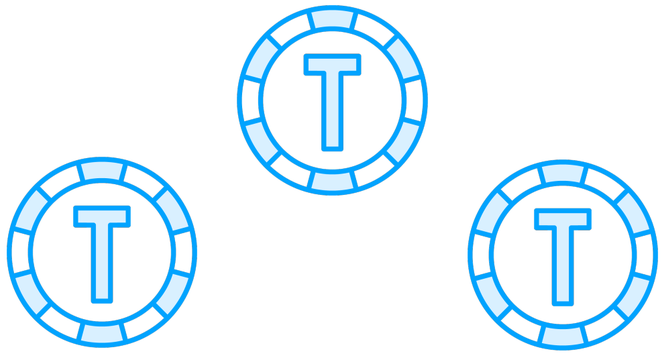
- Try different phrasings

Review results, identify weaknesses, seek feedback, test again, and repeat as needed

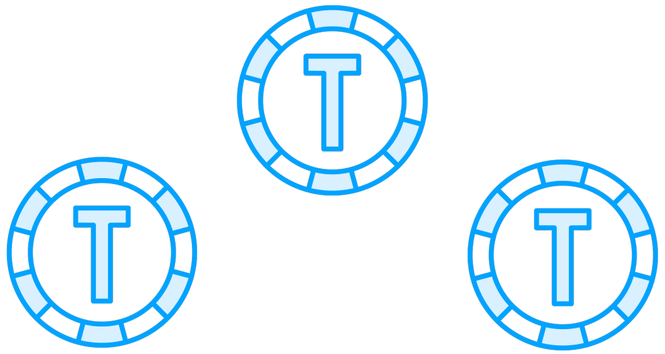
Tokenization

Toke what?

Toke what?



Tokenization



Process of splitting a text sequence

- Into smaller units
- Called "tokens"

Tokens are the smallest meaningful elements or symbols in a text

- Derived from the concept of linguistic tokens
- Units of language that convey meaning

Tokenization Example

The quick brown fox jumps over the lazy dog

Tokenizer

Learn about language model tokenization

OpenAI's large language models (sometimes referred to as GPT's) process text using **tokens**, which are common sequences of characters found in a set of text. The models learn to understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens.

You can use the tool below to understand how a piece of text might be tokenized by a language model, and the total count of tokens in that piece of text.

It's important to note that the exact tokenization process varies between models. This tool was built for early GPT-3 models and will not be accurate for newer models but can still be a useful when learning about how language models work.

GPT-3

The quick brown fox jumps over the lazy dog

Clear Show example

Tokens 9

Characters 43

The quick brown fox jumps over the lazy dog

TEXT TOKEN IDS

A helpful rule of thumb is that one token generally corresponds to ~4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens ~= 75 words).

Importance of Tokenization

Text processing

Analysis and statistics

Language modeling

Computational efficiency

Vocabulary mapping

Preprocessing and post- processing

The OpenAI Playground

Get started

Enter an instruction or select a preset, and watch the API respond with a completion that attempts to match the context or pattern you provided.

You can control which model completes your request by changing the model.

KEEP IN MIND

- Use good judgment when sharing outputs, and attribute them to your name or company. Learn more.
- Requests submitted to our API and Playground will not be used to train or improve future models. Learn more.
- Our default models' training data cuts off in 2021, so they may not have knowledge of current events.

Playground

Your presets | Save | View code | Share | ...

SYSTEM

You are a helpful assistant.

USER Enter a user message here.

Add message

Submit

Mode: Chat

Model: gpt-3.5-turbo

Temperature 1

Maximum length 256

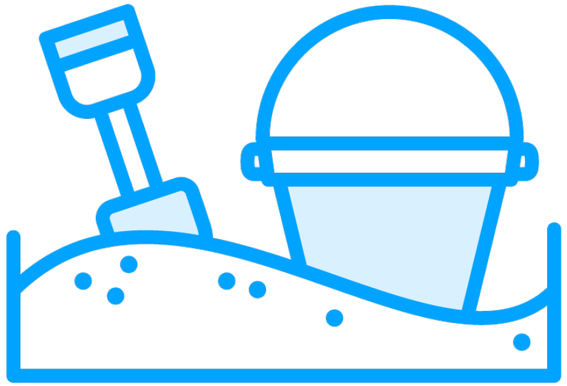
Stop sequences
Enter sequence and press Tab

Top P 1

Frequency penalty 0

Presence penalty 0

The OpenAI Playground



Web-based tool

- Allows you to test prompts
- Get familiar with the API

Test different models and settings

- Can save presets
- Share
- View code

The OpenAI Playground

Context is king!

Use system messages!

Prompting Best Practices

Prompting Best Practices a.k.a. Crafting Effective Prompts

Be clear and specific

Clearly articulate what you want

Avoid vague or ambiguous language

- *Tell me about dogs*

Specify the type of information or action you're looking for

- *What are the most common breeds of dogs?*

Use natural language

Write prompts in natural, conversational language that a human would use

Avoid overly technical or formal language

Frame questions as if you were asking a knowledgeable friend for help

Provide context

Include relevant context in your prompt

Explain the situation or background information

- If necessary

If the conversation has a history

- Refer back to previous messages to maintain the context

Ask open-ended questions

Encourage detailed responses by asking open-ended questions

- Can't be answered with a simple *yes* or *no*

For example, instead of asking

- *Is it raining?*

You can use

- *What's the weather like today?*

Break down complex questions

Consider breaking down complex questions into multiple smaller questions

- Easier for the AI model to provide accurate responses

For example, if I want to know about a city

- Ask separate questions about its history, attractions, and local cuisine
- Instead of
  - *Tell me everything about the city*

Avoid double negatives

Steer clear of double negatives or confusing phrasing that could lead to misinterpretation

For instance, instead of saying

- *Isn't it unlikely that this won't work?*

Simply ask

- *Will this work?*

Use relevant keywords

Identify and use keywords related to your topic

- Helps the model understand the subject matter better
- Helps provide more relevant responses

If you're asking about AI ethics, include terms like *machine learning*, *artificial intelligence*...

Test and iterate

Experiment with different prompts

- Analyze output
- Determine which ones yield the best results

AI models may respond differently to slight variations in wording

Don't hesitate to iterate and refine your prompts based on the AI's responses

Be polite and respectful

Maintain a polite and respectful tone in all your prompts

AI model should be treated with the same courtesy as a human

Avoid offensive or inappropriate language

Test with real users

Gather feedback from real users to fine-tune your prompts

They can provide valuable insights into the effectiveness of your questions

openai-cookbook

Public

Watch 790 Fork 8.1k Star 49.3k

main 5 branches 0 tags

Go to file Add file Code

simonpfish add function calling tags ccab1fe 2 hours ago 690 commits

.github	Polish PR template and rubric	yesterday
articles	Fix image links	4 hours ago
examples	Update title and reformat with prettier	2 hours ago
images	Update README to point to website (#732)	last month
.gitignore	Add description and rename Obtain_dataset, and throw error when fi...	last month
CONTRIBUTING.md	Polish PR template and rubric	yesterday
LICENSE	Create LICENSE (#136)	8 months ago
README.md	Update contributions process (#784)	5 days ago
registry.yaml	add function calling tags	2 hours ago

README.md

OpenAI Cookbook

> ✨ Navigate at cookbook.openai.com

Example code and guides for accomplishing common tasks with the OpenAI API. To run these examples, you'll

About

Examples and guides for using the OpenAI API

cookbook.openai.com

docs openai gpt-3 gpt-4 chatgpt

Readme

MIT license

Activity

49.3k stars

790 watching

8.1k forks

Report repository

Used by 1

@cram1k / tomfool

Contributors 148

README.md

OpenAI Cookbook

> ✨ Navigate at cookbook.openai.com

Example code and guides for accomplishing common tasks with the OpenAI API. To run these examples, you'll need an OpenAI account and associated API key (create a free account here).

Most code examples are written in Python, though the concepts can be applied in any language.

For other useful tools, guides and courses, check out these related resources from around the web.

Contributing

The OpenAI Cookbook is a community-driven resource. Whether you're submitting an idea, fixing a typo, adding a new guide, or improving an existing one, your contributions are greatly appreciated!

Before contributing, read through the existing issues and pull requests to see if someone else is already working on something similar. That way you can avoid duplicating efforts.

If there are examples or guides you'd like to see, feel free to suggest them on the issues page.

If you'd like to contribute new content, make sure to read through our contribution guidelines. We welcome high-quality submissions of new examples and guides, as long as they meet our criteria and fit within the scope of the cookbook.

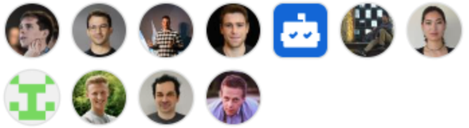
The contents of this repo are automatically rendered into cookbook.openai.com based on registry.yaml.

Open in GitHub Codespaces

Used by 1

@cram1k / **tomfool**

Contributors 148



+ 137 contributors

Popular

How to call functions with chat models

Jun 13, 2023 COMPLETIONS FUNCTIONS

How to count tokens with tiktoken

Dec 16, 2022 COMPLETIONS TIKTOKEN

Data preparation and analysis for chat model fine-tuning

Aug 22, 2023 COMPLETIONS TIKTOKEN

How to stream completions

Sep 2, 2022 COMPLETIONS TIKTOKEN

Question answering using embeddings-based search

Jun 10, 2022 COMPLETIONS EMBEDDINGS

How to format inputs to ChatGPT models

Mar 1, 2023 COMPLETIONS TIKTOKEN

Featured

How to build an agent with the Node.js SDK

Oct 5, 2023 COMPLETIONS FUNCTIONS

Related resources from around the web

Jan 20, 2023 COMPLETIONS EMBEDDINGS

Techniques to improve reliability

Sep 12, 2022 COMPLETIONS

How to work with large language models

Jan 20, 2023 COMPLETIONS

How to fine-tune chat models

Aug 22, 2023 COMPLETIONS

Evaluating Abstractive Summarization

Aug 16, 2023 COMPLETIONS EMBEDDINGS

All 115

Filter

How to build an agent with the Node.js SDK COMPLETIONS FUNCTIONS Oct 5, 2023

How to stream completions

Ted Sanders
Sep 2, 2022

Open in Github

By default, when you request a completion from the OpenAI, the entire completion is generated before being sent back in a single response.

If you're generating long completions, waiting for the response can take many seconds.

To get responses sooner, you can 'stream' the completion as it's being generated. This allows you to start printing or processing the beginning of the completion before the full completion is finished.

To stream completions, set `stream=True` when calling the chat completions or completions endpoints. This will return an object that streams back the response as data-only server-sent events. Extract chunks from the `delta` field rather than the `message` field.

Downsides

Note that using `stream=True` in a production application makes it more difficult to moderate the content of the completions, as partial completions may be more difficult to evaluate. This may have implications for approved usage.

Another small drawback of streaming responses is that the response no longer includes the `usage` field to tell you how many tokens were consumed. After receiving and combining all of the responses, you can calculate this yourself using `tiktoken`.

Six Best Practices for Prompt Engineering

Six Best Practices for Prompt Engineering

Write clear instructions

Provide reference text

Split complex tasks into simpler tasks

Give GPT time to "think"

Use external tools

Test changes systematically

Write Clear Instructions

Write Clear Instructions

Include details in your query to get more relevant answers

Ask the model to adopt a persona

Use delimiters to clearly indicate distinct parts of the input

Write Clear Instructions

Specify the steps required to complete a task

Provide examples

Specify the desired length of the output

Write clear instructions

Provide Reference Text

Provide Reference Text

Instruct the model to answer using a reference text

Instruct the model to answer with citations from a reference text

Provide reference text

Split Complex Tasks into Simpler Subtasks

Split Complex Tasks into Simpler Subtasks

Use intent classification to identify the most relevant instructions for a user query

For dialogue applications that require very long conversations, summarize or filter previous dialogue

Summarize long documents piecewise and construct a full summary recursively

Split complex tasks into simpler subtasks

Give GPTs Time to "Think"

Give GPTs Time to "Think"

Instruct the model to derive its conclusion through reasoning from fundamental principles

Employ inner monologue or a sequence of queries to conceal the model's reasoning process

Prompt the model to identify any previously overlooked content

Give GPTs time to "think"

Use External Tools and Test Changes Systematically

Use External Tools

GPT is not a one-size-fits-all solution

There are other tools that perform better in certain scenario

Use the tool that best solves a problem

Test Changes Systematically

You can’t improve what you don’t measure

A change may improve responses for certain prompts but may affect overall performance

Define a comprehensive test suite (eval)

A Few Final Words on Prompt Engineering

A Few Final Words on Prompt Engineering

- Use clear and concise language
- Provide examples of the desired output
- Break down complex prompts
- Provide feedback to the LLM and iterate

A Few Final Words on Prompt Engineering

- Use active voice
- Use positive language
- Use keywords that are related to your topic
- Use prompts that are consistent with the training data