Natural Language Processing Large Language Models Usage and Evaluation Patterns

Felipe Bravo-Marquez

December 5, 2023

Introduction

- Since the inception of Large Language Models, various patterns of use and evaluation of this technology have emerged.
- In this talk, we will try to organize these patterns and give a general overview of them.

Source:

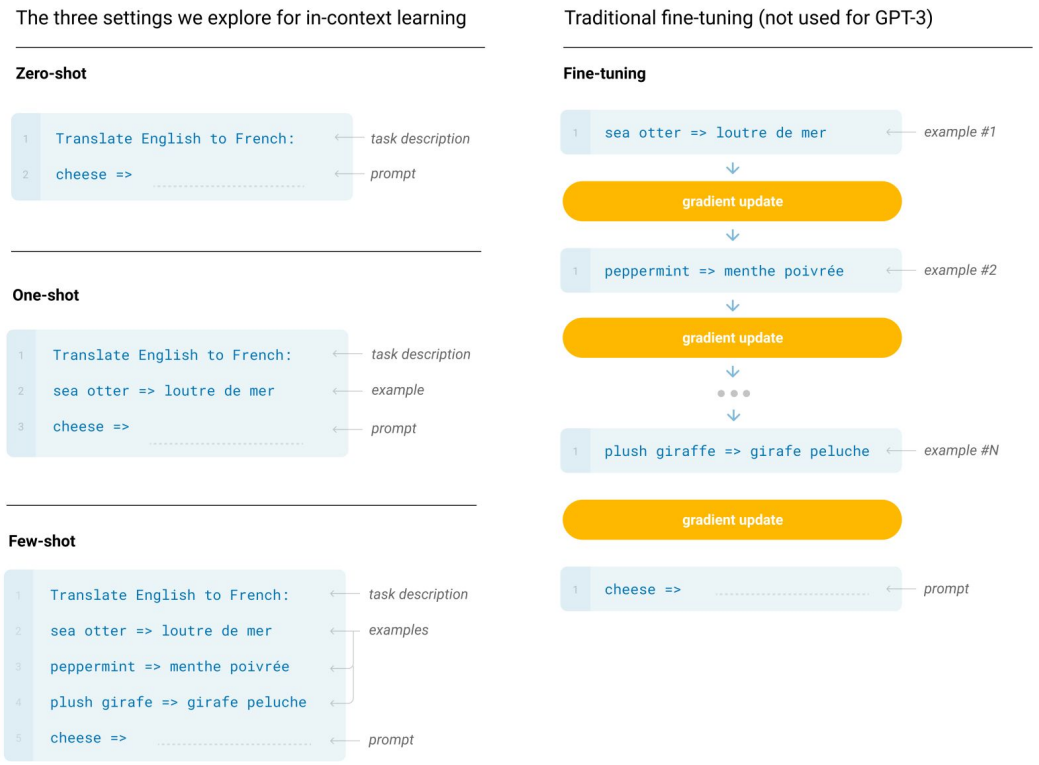
`https://www.masayume.it/img/masayume/Large-Language-Models.jpg`

Recap: What is an LLM

- Large Language Model: An autoregressive language model trained with a Transformer neural network on a large corpus (hundreds of bullions of tokens) and a large parameter space (billions) to predict the next word.
- It is usually later aligned to work as a user assistant using techniques such as Reinforcement Learning From Human Feedback [**?**] or supervised fine-tuning.
- Some are private (access via API or web browser): Google Bard, ChatGPT, etc.
- Others are open (model's weights can be downloaded): Llama, LLama2, Falcon, etc.

Zero-shot, One-shot, and Few-shot Learning

The most remarkable feature of these models is their few-shot, one-shot, zero-shot learning capabilities (also known as “in-context-learning”).



This means that they can learn new tasks without large amounts of human-annotated data.

Talk Overview

- Despite the recency of this technology, its adoption has been tremendous in many areas.
- Below, we propose a simple categorization of the ways in which LLMs are used and evaluated.
- These patterns will serve as the narrative backbone of this presentation.

Usage Patterns

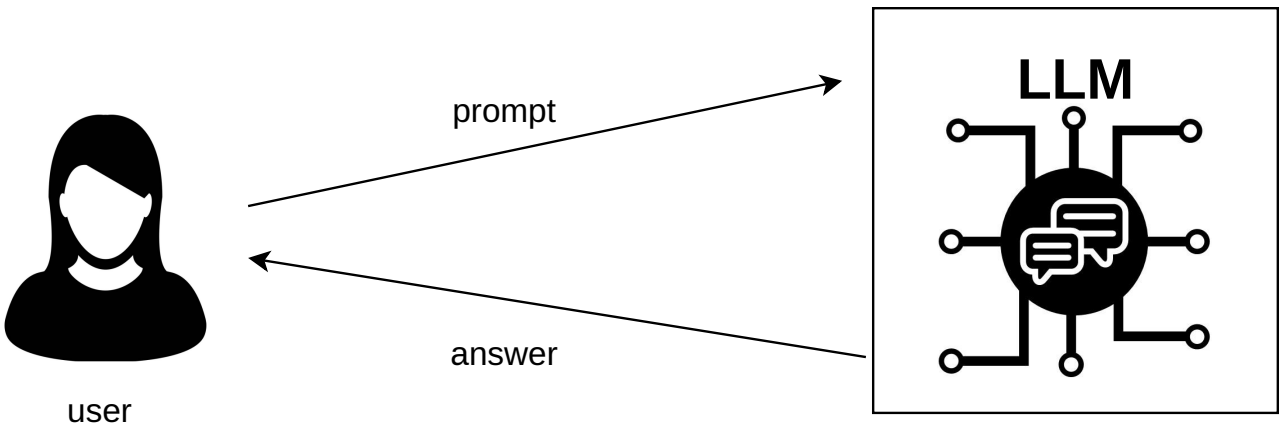
1. General-domain Assistant
2. Domain-specific Assistant
   - 2.1 Retrieval-augmented generation
   - 2.2 Fine-Tuning
3. LLM-based Applications
   - 3.1 API calls
   - 3.2 Autonomous Agents

Evaluation Patterns

- MTBench
- LLM Arena

Usage Pattern 1: General-domain Assistant

- In this pattern a user interacts with the LLM proving prompts as input and receiving a text as answer.
- The knowledge the LLM has access is limited to the corpus on which it was trained and the context given in the prompt.



Tasks

LLMs can solve many tasks with this pattern:

- Textual: Language understanding and common sense (e.g., rewriting, summarizing, translating, answering questions)
- Arithmetic: Mathematical reasoning (it can fail in many cases though)
- Visual: Multimodal reasoning involving pictures (GPT-4, Llava)
- Symbolic: Structured input such as programming languages

Source:

`https://twitter.com/IntuitMachine/status/1727079666001870877`.

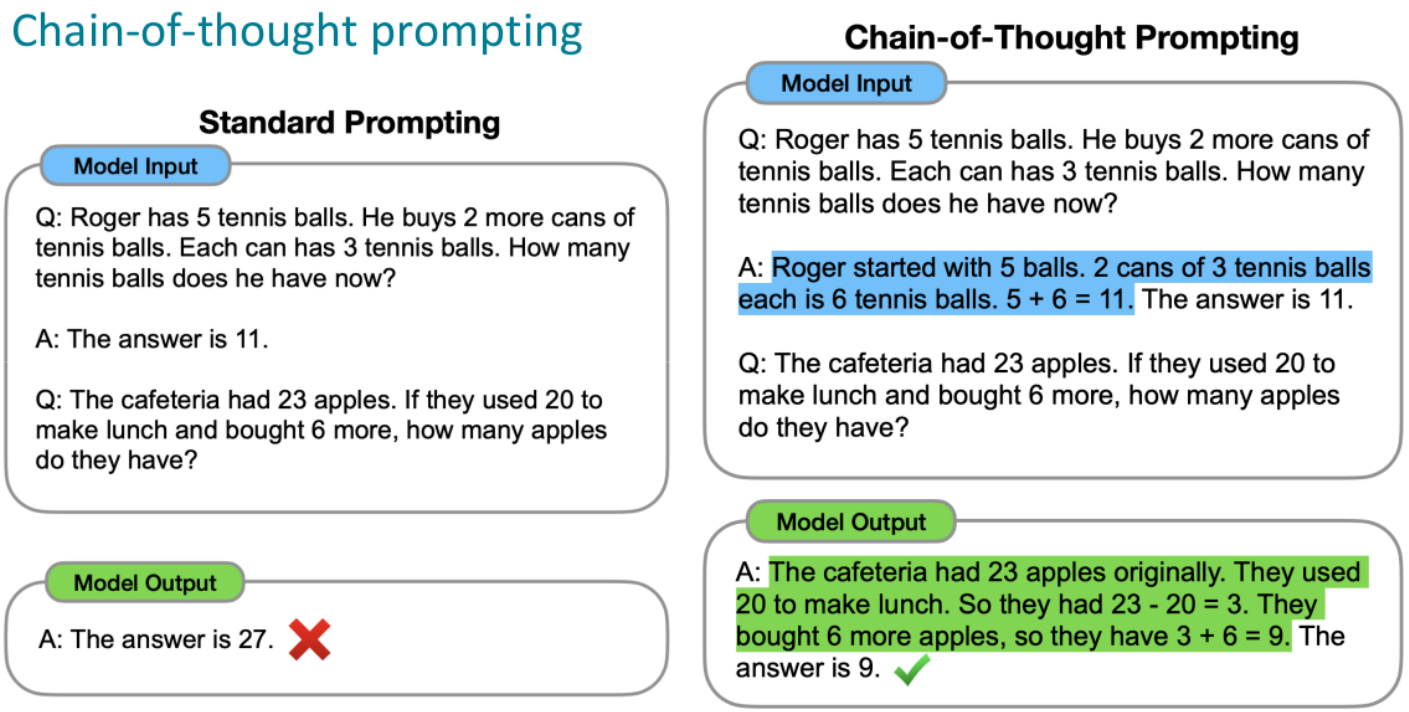
Prompt Engineering: Guiding the Language Model

Prompt engineering, often referred to as “Prompting,” is the discipline or “art” of crafting effective prompts to guide the Language Model (LM) towards generating accurate responses. Some common prompting guidelines:

- **Clarity and Conciseness**: Clearly articulate the prompt to minimize ambiguity and ensure the LM understands the task at hand.
- **Use of Specific Examples**: Provide concrete examples within the prompt to offer the LM context.
- **Role-based Prompts**: incorporating roles into the prompts (e.g., a tour guide, a teacher, a doctor).
- **Desired Output Specification**: Clearly define the desired format of the output (e.g, JSON, HTML, csv).

Chain-of-thought Prompting

- Chain-of-thought prompting is a simple mechanism for eliciting multi-step reasoning behavior in large language models.
- This method involves augmenting each exemplar in a few-shot prompt with a connected sequence of thoughts, creating a structured chain of logical steps. [**?**]

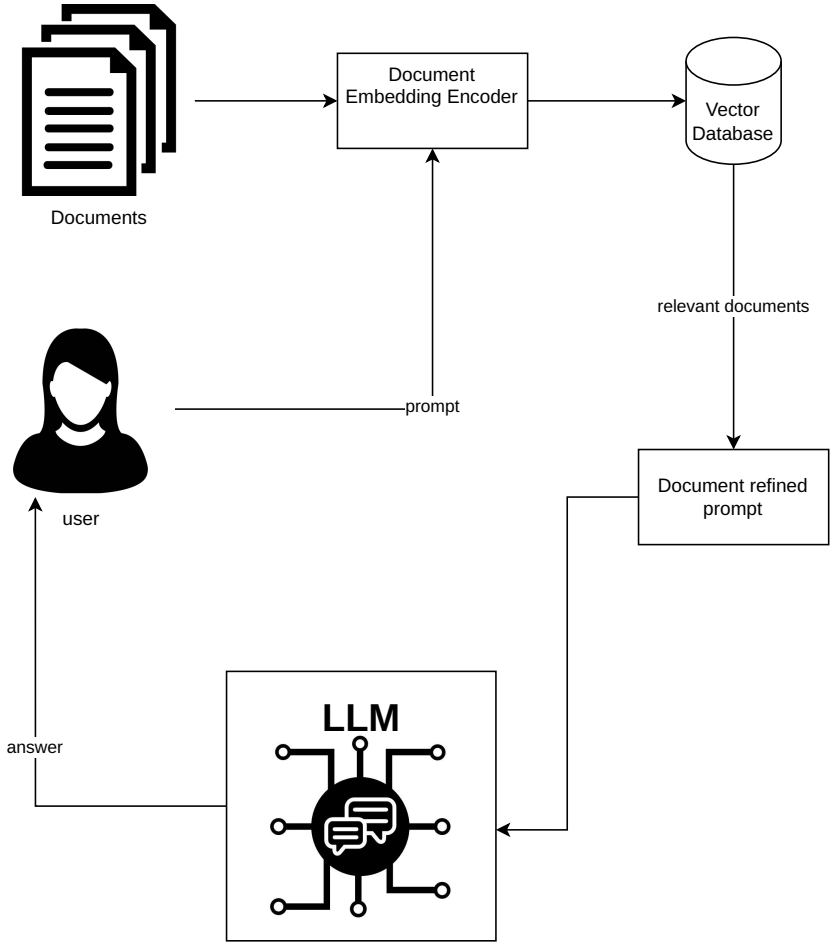


Usage Pattern 2: Domain-specific Assistant

- Idea: Incorporate domain-specific knowledge not covered in training (e.g., recent news, private documents).
- This is very common for companies developing chatbots with private documents or for creating more domain-specific chatbots.
- There are two main patterns to achieve this:
  1. Retrieval-Augmented Generation (Vector Databases)
  2. Fine-Tuning

Usage Pattern 2.1: Retrieval-Augmented Generation

Idea: Incorporate domain-specific knowledge into the query using information retrieval and document embeddings (i.e., densely encoded vectors that capture the semantic information of the document). [**?**].

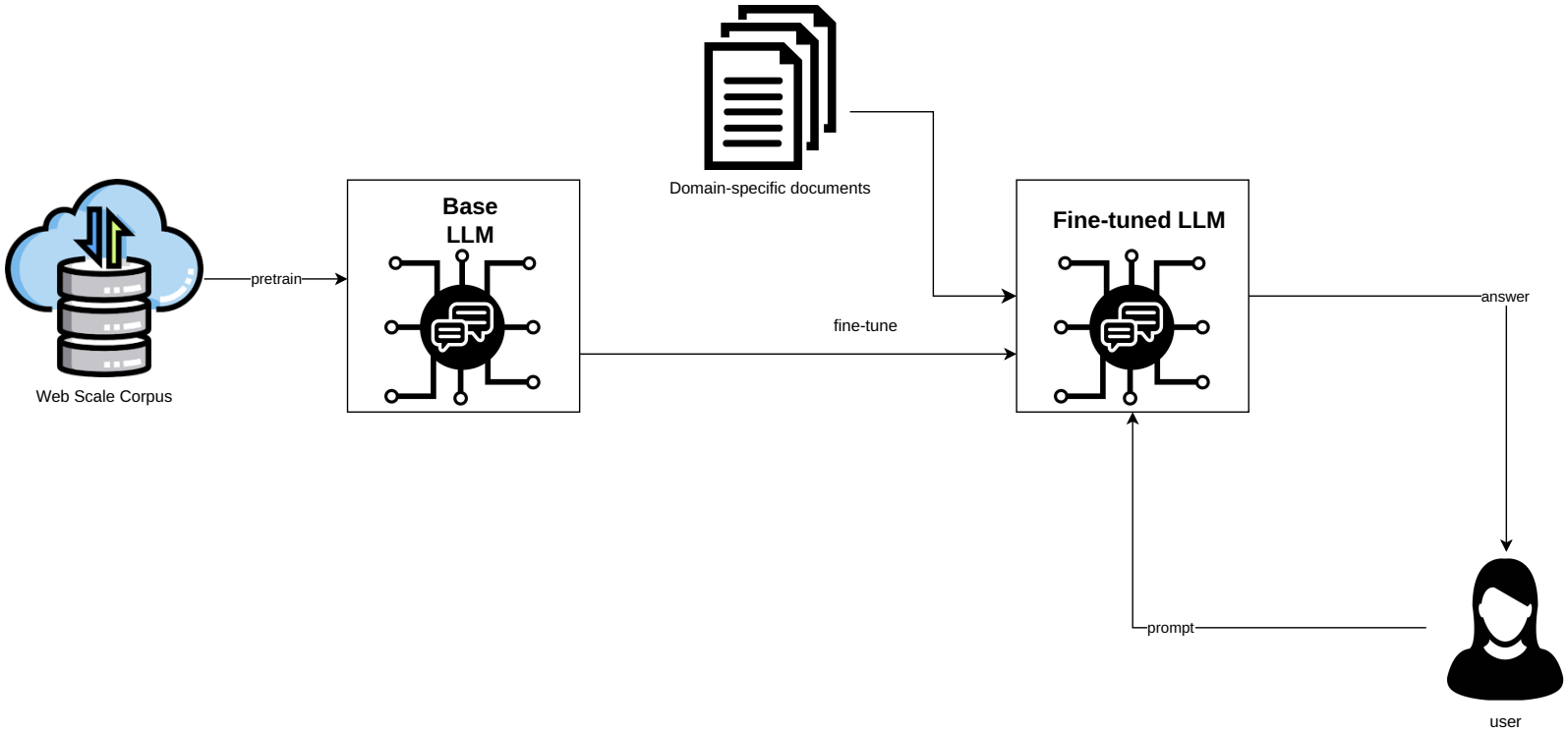


Retrieval-Augmented Generation Process

1. Encode all domain-specific documents using document embeddings and store them in a vector database.
   - OpenAI provides a vectorizer called (text2vec-openai), but there are many open source alternatives.
   - There are also many vector databases available, a popular one is **weaviate**.
2. Encode the prompt with the same vectorizer used to encode documents.
3. Use prompt embedding and the vector database to retrieve relevant documents based on similarity.
4. Create a refined prompt that includes a domain-specific role, the user prompt, and the retrieved documents as contexts.
5. Send the refined prompt to the LLM and return the response to the user.

Usage Pattern 2.2: Fine-Tuning

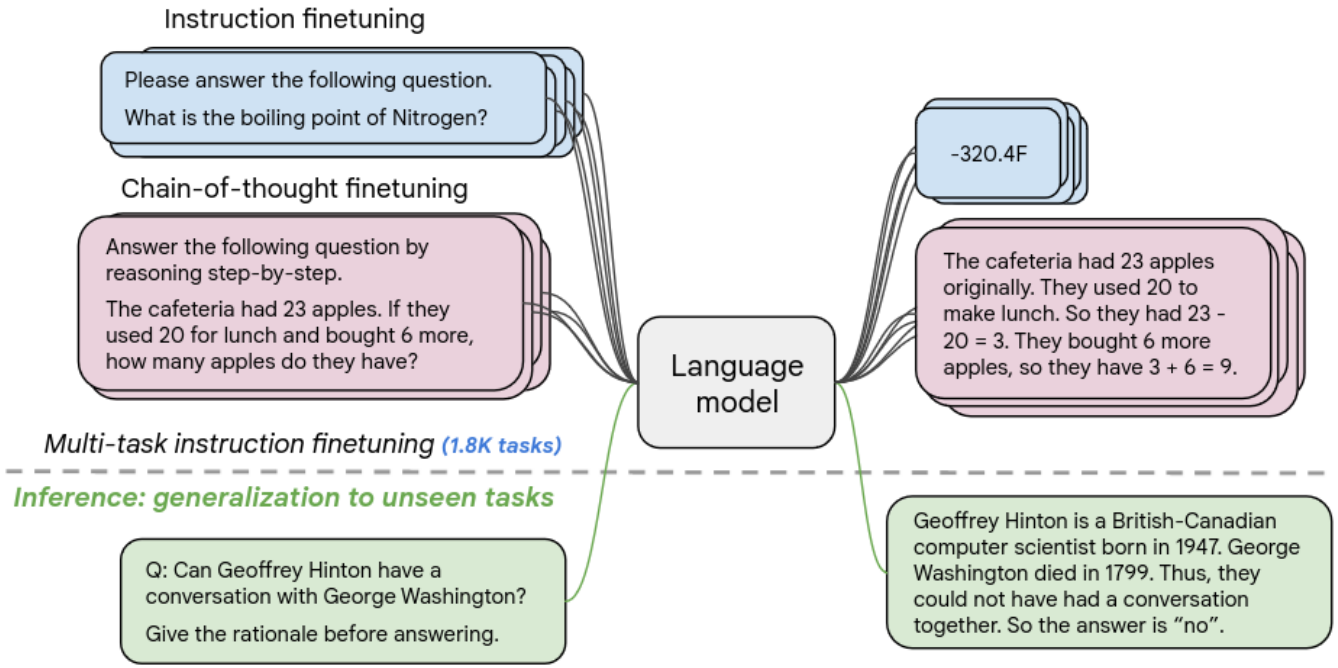
Idea: Incorporate domain-specific knowledge by fine-tuning a pre-trained LLM with the next-token prediction task over a domain-specific corpus and interact with the resulting LLM.



This can be computationally expensive unless some tricks are used.

Instruction Fine-tuning

- A more efficient way to fine-tune Large Language Models is Instruction Fine-Tuning [**?**].
- Idea: instead of fine-tuning the LLM with raw text with next token prediction, train it with pairs of prompts and user-aligned answers.
- Collect examples of (instruction, output) pairs across many tasks and finetune an LM.
- Evaluate on unseen tasks.



Datasets for Instruction Fine-Tuning

- Alpaca Data: 52k English instruction examples generated using OpenAI's text-davinci-003 with self-instruct.
- Evol-Instruct (Xu et al., April 2023): A rewritten set of 250k English instruction-response pairs based on the Alpaca data
- Vicuna ShareGPT: 70k English conversations shared by users and scraped from
- Baize data (Xu et al., April 2023)
- databricks-dolly-15k (Conover et al., April 2023)
- OpenAssistant Conversations (Köpf et al., April 2023)
- LIMA data (Zhou et al., May 2023).

source:

`https://nlpnewsletter.substack.com/p/instruction-tuning-vol-2`

Considerations for Instruction Tuning

- **Data Source:** How was the data obtained? Most datasets have been generated using ChatGPT. They may thus inherit biases of the source model or may be noisy.
- **Data Quality:** Was any filtering done to improve the quality of the generated data? In most cases, filtering is based on simple heuristics or a pre-trained model, which can result in noisy data.
- **Domain and Language Coverage:** Most datasets focus on general QA-style in English, but methods can be adapted for other domains or languages.
- **Number of Dialog Turns:** Single-turn datasets include a prompt and response. Consider multi-turn data for training a more conversational model.
- **License Terms:** Data from OpenAI models follows OpenAI terms, restricting use for competing models. Seek datasets with more permissive licenses to avoid legal complications.

source:
`https://nlpnewsletter.substack.com/p/instruction-tuning-vol-2`

Parameter Efficient Fine Tuning

- Lora, QLora
- https://blog.gopenai.com/paper-review-qlora-efficient-finetuning-of-quantized-llms-a3c857cd0cca

Token-Incrementation

- Lora, QLora
- https://blog.gopenai.com/paper-review-qlora-efficient-finetuning-of-quantized-llms-a3c857cd0cca

Low-Rank Adaptation (LoRA)

- Traditional model training is time and resource-intensive.
- LoRA is an effective reparameterization technique.
- Preserves pre-trained model weights.
- Introduces adjustable low-rank decomposition matrices.

Forward-pass equation:

$$h = W_0 x + \Delta W x = W_0 x + BAx, \quad B \in \mathbb{R}^{d \times r}, \quad A \in \mathbb{R}^{r \times d} \tag{1}$$

- h is the output, x is the input.
- W_0: Pre-trained weight matrix.
- B, A: Trainable low-rank decomposition matrices.

LoRA Implementation

Linear Layer

- For a linear layer with $W_0 \in \mathbb{R}^{d \times k}$:
  - d: Output dimension.
  - k: Input dimension.
- LoRA adds trainable matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$.
- r: Predefined low-rank.

Training Process

Freeze and Update

- During training:
  - W_0 remains frozen.
  - B and A receive gradient updates.
- $r \ll \min(d, k)$: Reduces memory consumption.
- Optimizer states only stored for B and A.

Benefits of LoRA

- Significant reduction in trainable parameters.
- Facilitates language model training with lower computational resources.
- Efficiently maintains the pre-trained model's weights.

Conclusion

Low-Rank Adaptation (LoRA)

- LoRA offers an efficient training approach.
- Preserving pre-trained weights.
- Notable reduction in the number of trainable parameters.
- Ideal for language models with resource constraints.

Applications

- LLMs can be embedded into any software via API calls. For example, a search engine (you.com)
- https://gptstore.ai/
- For example, PDF summarization software. You write software that first converts the PDF to raw text and then sends it to an LLM for summarization.
- Or a software for summarizing videoconferences. First the audio is transcribed and then summarized with an LLM.

Autonomous Agents

- Agents are a special kind of LLMs application in which the LLM serves as the reasoning and planning component of the software.
- agent in the sense of perceiving an environment and taking actions to achieve goals.

LLMBench and LLm Arena

- Standard NLP evaluation: human annotated gold-labels and metrics.
- LLMS are intrinsically multi-task and not easily evaluated with this approach.
- Machines evaluating machines??
- MT-bench (categories)
- HuggingFace Open LLM Leaderboard
- LLM Arena

Questions?

Thanks for your Attention!

References I